

PROYEK APLIKASI KOMPUTER



Saphira Nuria Salsabila
22305141050
Matematika B

DEPARTEMEN PENDIDIKAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN
ALAM
UNIVERSITAS NEGERI YOGYAKARTA
TAHUN AJARAN 2023/2024

November 30, 2023

BAB

Pendahuluan dan Pengenalan Cara Kerja EMT

Selamat datang! Ini adalah pengantar pertama ke Euler Math Toolbox (disingkat EMT atau Euler). EMT adalah sistem terintegrasi yang merupakan perpaduan kernel numerik Euler dan program komputer aljabar Maxima.

- Bagian numerik, GUI, dan komunikasi dengan Maxima telah dikembangkan oleh R. Grothmann, seorang profesor matematika di Universitas Eichstätt, Jerman. Banyak algoritma numerik dan pustaka software open source yang digunakan di dalamnya.
- Maxima adalah program open source yang matang dan sangat kaya untuk perhitungan simbolik dan aritmatika tak terbatas. Software ini dikelola oleh sekelompok pengembang di internet.
- Beberapa program lain (LaTeX, Povray, Tiny C Compiler, Python) dapat digunakan di Euler untuk memungkinkan perhitungan yang lebih cepat maupun tampilan atau grafik yang lebih baik.

Yang sedang Anda baca (jika dibaca di EMT) ini adalah berkas notebook di EMT. Notebook aslinya bawaan EMT (dalam bahasa Inggris) dapat dibuka melalui menu File, kemudian pilih "Open Tutorias and Example", lalu pilih file "00 First Steps.en". Perhatikan, file notebook EMT memiliki ekstensi ".en". Melalui notebook ini Anda akan belajar menggunakan software Euler untuk menyelesaikan berbagai masalah matematika.

Panduan ini ditulis dengan Euler dalam bentuk notebook Euler, yang berisi teks (deskriptif), baris-baris perintah, tampilan hasil perintah (numerik, ekspresi matematika, atau gambar/plot), dan gambar yang disisipkan dari file gambar.

Untuk menambah jendela EMT, Anda dapat menekan [F11]. EMT akan menampilkan jendela grafik di layar desktop Anda. Tekan [F11] lagi untuk kembali ke tata letak favorit Anda. Tata letak disimpan untuk sesi berikutnya.

Anda juga dapat menggunakan [Ctrl]+[G] untuk menyembunyikan jendela grafik. Selanjutnya Anda dapat beralih antara grafik dan teks dengan tombol [TAB].

Seperti yang Anda baca, notebook ini berisi tulisan (teks) berwarna hijau, yang dapat Anda edit dengan mengklik kanan teks atau tekan menu Edit -> Edit Comment atau tekan [F5], dan juga baris perintah EMT yang ditandai dengan ">" dan berwarna merah. Anda dapat menyisipkan baris perintah baru dengan cara menekan tiga tombol bersamaan: [Shift]+[Ctrl]+[Enter].

Komentar atau teks penjelasan dapat berisi beberapa "markup" dengan sintaks sebagai berikut.

- * Judul
- ** Sub-Judul
- latex: $F(x) = \int_a^x f(t) dt$
- mathjax: $\frac{x^2-1}{x-1} = x + 1$
- maxima: 'integrate(x^3,x) = integrate(x^3,x) + C
- http://www.euler-math-toolbox.de
- See: http://www.google.de | Google
- image: hati.png
- ---

Hasil sintaks-sintaks di atas (tanpa diawali tanda strip) adalah sebagai berikut.

Judul

Sub-Judul

$$F(x) = \int_a^x f(t) dt$$

$$\frac{x^2 - 1}{x - 1} = x + 1$$

maxima: 'integrate(x^3,x) = integrate(x^3,x) + C

http://www.euler-math-toolbox.de

See: http://www.google.de | Google

image: hati.png

Gambar diambil dari folder images di tempat file notebook berada dan tidak dapat dibaca dari Web. Untuk "See:", tautan (URL)web lokal dapat digunakan.

Paragraf terdiri atas satu baris panjang di editor. Pergantian baris akan memulai baris baru. Paragraf harus dipisahkan dengan baris kosong.

>/ baris perintah diawali dengan >, komentar (keterangan) diawali dengan /

Mari kita tunjukkan cara menggunakan EMT sebagai kalkulator yang sangat canggih. EMT berorientasi pada baris perintah. Anda dapat menuliskan satu atau lebih perintah dalam satu baris perintah. Setiap perintah harus diakhiri dengan koma atau titik koma.

- Titik koma menyembunyikan output (hasil) dari perintah.
- Sebuah koma mencetak hasilnya.
- Setelah perintah terakhir, koma diasumsikan secara otomatis (boleh tidak ditulis).

Dalam contoh berikut, kita mendefinisikan variabel r yang diberi nilai 1,25. Output dari definisi ini adalah nilai variabel. Tetapi karena tanda titik koma, nilai ini tidak ditampilkan. Pada kedua perintah di belakangnya, hasil kedua perhitungan tersebut ditampilkan.

```
>r=1.25; pi*r^2, 2*pi*r
```

4.90873852123

7.85398163397

Latihan untuk Anda

- Sisipkan beberapa baris perintah baru
- Tulis perintah-perintah baru untuk melakukan suatu perhitungan yang Anda inginkan, boleh menggunakan variabel, boleh tanpa variabel.

```
> 25*4
```

100

```
> sin(135°)
```

0.707106781187

```
> sin(30°), cos(90°), sin(pi)
```

0.5

0

0

Penjelasan :

Cara untuk menyisipkan perintah yaitu dengan meneklik Ctrl+Shift+Enter pada tempat yang diinginkan, lalu mengetik perintah sesuai yang diinginkan.

Beberapa catatan yang harus Anda perhatikan tentang penulisan sintaks perintah EMT.

- Pastikan untuk menggunakan titik desimal, bukan koma desimal untuk bilangan!
- Gunakan * untuk perkalian dan ^ untuk eksponen (pangkat).
- Seperti biasa, * dan / bersifat lebih kuat daripada + atau -.
- ^ mengikat lebih kuat dari *, sehingga pi * r ^ 2 merupakan rumus luas lingkaran.
- Jika perlu, Anda harus menambahkan tanda kurung, seperti pada 2 ^ (2 ^ 3).

Perintah $r = 1.25$ adalah menyimpan nilai ke variabel di EMT. Anda juga dapat menulis $r := 1.25$ jika mau. Anda dapat menggunakan spasi sesuka Anda.

Anda juga dapat mengakhiri baris perintah dengan komentar yang diawali dengan dua garis miring (//).

```
>r := 1.25 // Komentar: Menggunakan := sebagai ganti =
```

1.25

Argumen atau input untuk fungsi ditulis di dalam tanda kurung.

```
>sin(45°), cos(pi), log(sqrt(E))
```

0.707106781187

-1

0.5

Seperti yang Anda lihat, fungsi trigonometri bekerja dengan radian, dan derajat dapat diubah dengan °. Jika keyboard Anda tidak memiliki karakter derajat tekan [F7], atau gunakan fungsi deg() untuk mengonversi.

EMT menyediakan banyak sekali fungsi dan operator matematika. Hampir semua fungsi matematika sudah tersedia di EMT. Anda dapat melihat daftar lengkap fungsi-fungsi matematika di EMT pada berkas Referensi (klik menu Help -> Reference)

Untuk membuat rangkaian komputasi lebih mudah, Anda dapat merujuk ke hasil sebelumnya dengan "%". Cara ini sebaiknya hanya digunakan untuk merujuk hasil perhitungan dalam baris perintah yang sama.

```
>(sqrt(5)+1)/2, %^2-%+1 // Memeriksa solusi x^2-x+1=0
```

1 . 61803398875

2

Latihan untuk Anda

- Buka berkas Reference dan baca fungsi-fungsi matematika yang tersedia di EMT.
 - Sisipkan beberapa baris perintah baru.
 - Lakukan contoh-contoh perhitungan menggunakan fungsi-fungsi matematika di EMT.
-

```
> gcd(343, 9)
```

1

```
> factor(gcd(1000, 25))
```

[5, 5]

```
> lcm(1024, 8)
```

1024

```
> lcm(13, 17)
```

221

Satuan

EMT dapat mengubah unit satuan menjadi sistem standar internasional (SI). Tambahkan satuan di belakang angka untuk konversi sederhana.

```
> 1miles // 1 mil = 1609,344 m
```

1609.344

Beberapa satuan yang sudah dikenal di dalam EMT adalah sebagai berikut. Semua unit diakhiri dengan tanda dolar (\$), namun boleh tidak perlu ditulis dengan mengaktifkan easy-units.

```

kilometer$:=1000;
km$:=kilometer$;
cm$:=0.01;
mm$:=0.001;
minute$:=60;
min$:=minute$;
minutes$:=minute$;
hour$:=60*minute$;
h$:=hour$;
hours$:=hour$;
day$:=24*hour$;
days$:=day$;
d$:=day$;
year$:=365.2425*day$;
years$:=year$;
y$:=year$;
inch$:=0.0254;
in$:=inch$;
feet$:=12*inch$;
foot$:=feet$;
ft$:=feet$;
yard$:=3*feet$;
yards$:=yard$;
yd$:=yard$;
mile$:=1760*yard$;
miles$:=mile$;
kg$:=1;
sec$:=1;
ha$:=10000;
Ar$:=100;
Tagwerk$:=3408;
Acre$:=4046.8564224;
pt$:=0.376mm;
```

Untuk konversi ke dan antar unit, EMT menggunakan operator khusus, yakni ->.

```
>4km -> miles, 4inch -> " mm"
```

2 . 48548476895

101 . 6 mm

Format Tampilan Nilai

Akurasi internal untuk nilai bilangan di EMT adalah standar IEEE, sekitar 16 digit desimal. Aslinya, EMT tidak mencetak semua digit suatu bilangan. Ini untuk menghemat tempat dan agar terlihat lebih baik. Untuk mengatramilan satu bilangan, operator berikut dapat digunakan.

```
>pi
```

```
3.14159265359
```

```
>longest pi
```

```
3.141592653589793
```

```
>long pi
```

```
3.14159265359
```

```
>short pi
```

```
3.1416
```

```
>shortest pi
```

```
3.1
```

```
>fraction pi
```

```
312689/99532
```

```
>short 1200*1.03^10, long E, longest pi
```

```
1612.7
```

```
2.71828182846
```

```
3.141592653589793
```

Format aslinya untuk menampilkan nilai menggunakan sekitar 10 digit. Format tampilan nilai dapat diatur secara global atau hanya untuk satu nilai.

Anda dapat mengganti format tampilan bilangan untuk semua perintah selanjutnya. Untuk mengembalikan ke format aslinya dapat digunakan perintah "deformat" atau "reset".

```
>longestformat; pi, defformat; pi
```

3.141592653589793

3.14159265359

Kernel numerik EMT bekerja dengan bilangan titik mengambang (floating point) dalam presisi ganda IEEE (berbeda dengan bagian simbolik EMT). Hasil numerik dapat ditampilkan dalam bentuk pecahan.

```
>1/7+1/4, fraction %
```

0.392857142857

11/28

Perintah Multibaris

Perintah multi-baris membentang di beberapa baris yang terhubung dengan "..." di setiap akhir baris, kecuali baris terakhir. Untuk menghasilkan tanda pindah baris tersebut, gunakan tombol [Ctrl]+[Enter]. Ini akan menyambung perintah ke baris berikutnya dan menambahkan "..." di akhir baris sebelumnya. Untuk menggabungkan suatu baris ke baris sebelumnya, gunakan [Ctrl]+[Backspace].

Contoh perintah multi-baris berikut dapat dijalankan setiap kali kursor berada di salah satu barisnya. Ini juga menunjukkan bahwa ... harus berada di akhir suatu baris meskipun baris tersebut memuat komentar.

```
>a=4; b=15; c=2; // menyelesaikan a*x^2+b*x+c=0 secara manual ...
>D=sqrt (b^2/(a^2*4)-c/a); ...
>-b/(2*a) + D, ...
>-b/(2*a) - D
```

-0.138444501319

-3.61155549868

Menampilkan Daftar Variabel

Untuk menampilkan semua variabel yang sudah pernah Anda definisikan sebelumnya (dan dapat dilihat kembali nilainya), gunakan perintah "listvar".

```
>listvar
```

r	1.25
a	4
b	15
c	2
D	1.73655549868123

Perintah listvar hanya menampilkan variabel buatan pengguna. Dimungkinkan untuk menampilkan variabel lain, dengan menambahkan string termuat di dalam nama variabel yang diinginkan.

Perlu Anda perhatikan, bahwa EMT membedakan huruf besar dan huruf kecil. Jadi variabel "d" berbeda dengan variabel "D".

Contoh berikut ini menampilkan semua unit yang diakhiri dengan "m" dengan mencari semua variabel yang berisi "m\$".

```
>listvar m$
```

km\$	1000
cm\$	0.01
mm\$	0.001
nm\$	1853.24496
gram\$	0.001
m\$	1
hquantum\$	6.62606957e-34
atm\$	101325

Untuk menghapus variabel tanpa harus memulai ulang EMT gunakan perintah "remvalue".

```
>remvalue a,b,c,D
>D
```

```
Variable D not found!
Error in:
D ...
^
```

Menampilkan Panduan

Untuk mendapatkan panduan tentang penggunaan perintah atau fungsi di EMT, buka jendela panduan dengan menekan [F1] dan cari fungsinya. Anda juga dapat mengklik dua kali pada fungsi yang tertulis di baris perintah atau di teks untuk membuka jendela panduan.

Coba klik dua kali pada perintah "intrandom" berikut ini!

```
>intrandom(10, 6)
```

```
[4, 2, 6, 2, 4, 2, 3, 2, 2, 6]
```

Di jendela panduan, Anda dapat mengklik kata apa saja untuk menemukan referensi atau fungsi.

Misalnya, coba klik kata "random" di jendela panduan. Kata tersebut boleh ada dalam teks atau di bagian "See:" pada panduan. Anda akan menemukan penjelasan fungsi "random", untuk menghasilkan bilangan acak berdistribusi uniform antara 0,0 dan 1,0. Dari panduan untuk "random" Anda dapat menampilkan panduan untuk fungsi "normal", dll.

```
>random(10)
```

```
[0.270906, 0.704419, 0.217693, 0.445363, 0.308411, 0.914541, 0.1935
```

$$0.463387, 0.095153, 0.595017]$$

```
>normal(10)
```

```
[-0.495418, 1.6463, -0.390056, -1.98151, 3.44132, 0.308178, -0.7334
```

$$-0.526167, 1.10018, 0.108453]$$

Matriks dan Vektor

EMT merupakan suatu aplikasi matematika yang mengerti "bahasa matriks". Artinya, EMT menggunakan vektor dan matriks untuk perhitungan-perhitungan tingkat lanjut. Suatu vektor atau matriks dapat didefinisikan dengan tanda kurung siku. Elemen-elemennya dituliskan di dalam tanda kurung siku, antar elemen dalam satu baris dipisahkan oleh koma(),, antar baris dipisahkan oleh titik koma();.

Vektor dan matriks dapat diberi nama seperti variabel biasa.

```
>v=[4,5,6,3,2,1]
```

```
[4, 5, 6, 3, 2, 1]
```

```
>A=[1,2,3;4,5,6;7,8,9]
```

1	2	3
4	5	6
7	8	9

Karena EMT mengerti bahasa matriks, EMT memiliki kemampuan yang sangat canggih untuk melakukan perhitungan matematis untuk masalah-masalah aljabar linier, statistika, dan optimisasi.

Vektor juga dapat didefinisikan dengan menggunakan rentang nilai dengan interval tertentu menggunakan tanda titik dua (:), seperti contoh berikut ini.

```
>c=1:5
```

```
[1, 2, 3, 4, 5]
```

```
>w=0:0.1:1
```

```
[0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]
```

```
>mean (w^2)
```

```
0.35
```

Bilangan Kompleks

EMT juga dapat menggunakan bilangan kompleks. Tersedia banyak fungsi untuk bilangan kompleks di EMT. Bilangan imaginer

$$i = \sqrt{-1}$$

dituliskan dengan huruf I (huruf besar I), namun akan ditampilkan dengan huruf i (i kecil).

```
re(x) : bagian riil pada bilangan kompleks x.  
im(x) : bagian imaginer pada bilangan kompleks x.  
complex(x) : mengubah bilangan riil x menjadi bilangan kompleks.  
conj(x) : Konjugat untuk bilangan bilangan kompleks x.  
arg(x) : argumen (sudut dalam radian) bilangan kompleks x.  
real(x) : mengubah x menjadi bilangan riil.
```

Apabila bagian imaginer x terlalu besar, hasilnya akan menampilkan pesan kesalahan.

```
>sqrt (-1) // Error!  
>sqrt (complex (-1))
```

```
>z=2+3*I, re(z), im(z), conj(z), arg(z), deg(arg(z)), deg(arctan(3/2))
```

```
2+3i
2
3
2-3i
0.982793723247
56.309932474
56.309932474
```

```
>deg(arg(I)) // 90°
```

```
90
```

```
>sqrt(-1)
```

```
Floating point error!
Error in sqrt
Error in:
sqrt(-1) ...
^
```

```
>sqrt(complex(-1))
```

```
0+1i
```

EMT selalu menganggap semua hasil perhitungan berupa bilangan riil dan tidak akan secara otomatis mengubah ke bilangan kompleks.

Jadi akar kuadrat -1 akan menghasilkan kesalahan, tetapi akar kuadrat kompleks didefinisikan untuk bidang koordinat dengan cara seperti biasa. Untuk mengubah bilangan riil menjadi kompleks, Anda dapat menambahkan 0i atau menggunakan fungsi "complex".

```
>complex(-1), sqrt(%)
```

```
-1+0i
0+1i
```

EMT dapat melakukan perhitungan matematika simbolis (eksak) dengan bantuan software Maxima. Software Maxima otomatis sudah terpasang di komputer Anda ketika Anda memasang EMT. Meskipun demikian, Anda dapat juga memasang software Maxima tersendiri (yang terpisah dengan instalasi Maxima di EMT).

Pengguna Maxima yang sudah mahir harus memperhatikan bahwa terdapat sedikit perbedaan dalam sintaks antara sintaks asli Maxima dan sintaks ekspresi simbolik di EMT.

Untuk melakukan perhitungan matematika simbolis di EMT, awali perintah Maxima dengan tanda "&". Setiap ekspresi yang dimulai dengan "&" adalah ekspresi simbolis dan dikerjakan oleh Maxima.

```
>& (a+b)^2
```

$$(b + a)^2$$

```
>&expand( (a+b)^2), &factor(x^2+5*x+6)
```

$$b^2 + 2ab + a^2$$

$$(x + 2)(x + 3)$$

```
>&solve(a*x^2+b*x+c,x) // rumus abc
```

$$[x = \frac{(-\sqrt{b^2 - 4ac}) - b}{2a}, x = \frac{\sqrt{b^2 - 4ac} - b}{2a}]$$

```
>&(a^2-b^2)/(a+b), &ratsimp(%)// ratsimp menyederhanakan bentuk pecahan
```

$$\begin{array}{r} 2 \quad 2 \\ a - b \\ \hline b + a \end{array}$$

$$a - b$$

```
>10! // nilai faktorial (modus EMT)
```

3628800

```
>&10! //nilai faktorial (simbolik dengan Maxima)
```

3628800

Untuk menggunakan perintah Maxima secara langsung (seperti perintah pada layar Maxima) awali perintahnya dengan tanda ":" pada baris perintah EMT. Sintaks Maxima disesuaikan dengan sintaks EMT (disebut "modus kompatibilitas").

```
>factor(1000) // mencari semua faktor 1000 (EMT)
```

[2, 2, 2, 5, 5, 5]

```
>::: factor(1000) // faktorisasi prima 1000 (dengan Maxima)
```

$$\begin{array}{r} 3 \quad 3 \\ 2 \quad 5 \end{array}$$

```
>::: factor(20!)
```

$$\begin{array}{r} 18 \quad 8 \quad 4 \quad 2 \\ 2 \quad 3 \quad 5 \quad 7 \quad 11 \quad 13 \quad 17 \quad 19 \end{array}$$

Jika Anda sudah mahir menggunakan Maxima, Anda dapat menggunakan sintaks asli perintah Maxima dengan menggunakan tanda "://" untuk mengawali setiap perintah Maxima di EMT. Perhatikan, harus ada spasi antara "://" dan perintahnya.

```
>::: binomial(5,2); // nilai C(5,2)
```

10

```
>::: binomial(m,4); // C(m,4)=m!/ (4! (m-4) !)
```

$$\frac{(m - 3)(m - 2)(m - 1)m}{24}$$

```
>::: trigexpand(cos(x+y)); // rumus cos(x+y)=cos(x) cos(y)-sin(x) sin(y)
```

$$\cos(x) \cos(y) - \sin(x) \sin(y)$$

```
>::: trigexpand(sin(x+y));
```

$$\cos(x) \sin(y) + \sin(x) \cos(y)$$

```
>::: trigsimp(((1-sin(x)^2)*cos(x))/cos(x)^2+tan(x)*sec(x)^2) //menyederhanakan
```

$$\frac{\sin^4(x) + \cos^4(x)}{\cos^3(x)}$$

Untuk menyimpan ekspresi simbolik ke dalam suatu variabel digunakan tanda "&=".

```
>p1 &= (x^3+1) / (x+1)
```

$$\begin{array}{r} 3 \\ x + 1 \\ \hline x + 1 \end{array}$$

```
>&ratsimp(p1)
```

$$\begin{array}{r} 2 \\ x - x + 1 \end{array}$$

Untuk mensubstitusikan suatu nilai ke dalam variabel dapat digunakan perintah "with".

```
>&p1 with x=3 // (3^3+1) / (3+1)
```

7

```
>&p1 with x=a+b, &ratsimp(%) //substitusi dengan variabel baru
```

$$\begin{array}{r} 3 \\ (b + a) + 1 \\ \hline b + a + 1 \end{array}$$

$$\begin{array}{c} 2 \quad \quad \quad 2 \\ b + (2 a - 1) b + a - a + 1 \end{array}$$

```
>&diff(p1,x) //turunan p1 terhadap x
```

$$\begin{array}{r} 2 \quad \quad \quad 3 \\ 3x \quad \quad x + 1 \\ \hline x + 1 \quad \quad \quad 2 \\ (x + 1) \end{array}$$

```
>&integrate(p1,x) // integral p1 terhadap x
```

$$\begin{array}{r} 3 \quad \quad \quad 2 \\ 2x^3 - 3x^2 + 6x \\ \hline 6 \end{array}$$

Tampilan Matematika Simbolik dengan LaTeX

Anda dapat menampilkan hasil perhitungan simbolik secara lebih bagus menggunakan LaTeX. Untuk melakukan hal ini, tambahkan tanda dolar (\$) di depan tanda & pada setiap perintah Maxima.

Perhatikan, hal ini hanya dapat menghasilkan tampilan yang diinginkan apabila komputer Anda sudah terpasang software LaTeX.

```
>$& (a+b)^2
>$&expand((a+b)^2), $&factor(x^2+5*x+6)
>$&solve(a*x^2+b*x+c,x) // rumus abc
>$& (a^2-b^2)/(a+b), $&ratsimp(%)
```

Selamat Belajar dan Berlatih!

Baik, itulah sekilas pengantar penggunaan software EMT. Masih banyak kemampuan EMT yang akan Anda pelajari dan praktikkan.

Sebagai latihan untuk memperlancar penggunaan perintah-perintah EMT yang sudah dijelaskan di atas, silakan Anda lakukan hal-hal sebagai berikut.

- Carilah soal-soal matematika dari buku-buku Matematika.
- Tambahkan beberapa baris perintah EMT pada notebook ini.
- Selesaikan soal-soal matematika tersebut dengan menggunakan EMT.

Pilih soal-soal yang sesuai dengan perintah-perintah yang sudah dijelaskan dan dicontohkan di atas.

Terdapat 10 mahasiswa balon pengurus BEM. Akan dipilih 3 orang mahasiswa sebagai ketua, sekretaris, dan bendahara. Jika semua balon memenuhi kriteria untuk dipilih, ada berapa susunan kepengurusan yang mungkin?

```
> 10! / ((10-3) !)
```

720

BAB

EMT untuk Perhitungan Aljabar

Pada notebook ini Anda belajar menggunakan EMT untuk melakukan berbagai perhitungan terkait dengan materi atau topik dalam Aljabar. Kegiatan yang harus Anda lakukan adalah sebagai berikut:

- Membaca secara cermat dan teliti notebook ini;
- Menerjemahkan teks bahasa Inggris ke bahasa Indonesia;
- Mencoba contoh-contoh perhitungan (perintah EMT) dengan cara meng-ENTER setiap perintah EMT yang ada (pindahkan kursor ke baris perintah)
- Jika perlu Anda dapat memodifikasi perintah yang ada dan memberikan keterangan/penjelasan tambahan terkait hasilnya.
- Menyisipkan baris-baris perintah baru untuk mengerjakan soal-soal Aljabar dari file PDF yang saya berikan;
- Memberi catatan hasilnya.
- Jika perlu tuliskan soalnya pada teks notebook (menggunakan format LaTeX).
- Gunakan tampilan hasil semua perhitungan yang eksak atau simbolik dengan format LaTeX. (Seperti contoh-contoh pada notebook ini.)

Contoh pertama

Menyederhanakan bentuk aljabar:

$$6x^{-3}y^5 \times -7x^2y^{-9}$$

```
> $& 6*x^(-3)*y^5*-7*x^2*y^(-9)
```

$$-\frac{42}{x y^4}$$

$$(6x^{-3} + y^5)(-7x^2 - y^{-9})$$

```
> $&showev ('expand( (6*x^(-3)+y^5)*(-7*x^2-y^(-9)) ))
```

$$\text{expand} \left(\left(-\frac{1}{y^9} - 7x^2 \right) \left(y^5 + \frac{6}{x^3} \right) \right) = -7x^2 y^5 - \frac{1}{y^4} - \frac{6}{x^3 y^9} - \frac{42}{x}$$

Baris Perintah

Baris perintah Euler terdiri dari satu atau beberapa perintah Euler diikuti dengan titik koma ";" atau koma ",". Titik koma mencegah pencetakan hasilnya. Koma setelah perintah terakhir dapat dihilangkan.

Baris perintah berikut hanya akan mencetak hasil ekspresi, bukan tugas atau perintah format.

```
>r:=2; h:=4; pi*r^2*h/3
```

16.7551608191

Perintah harus dipisahkan dengan yang kosong. Baris perintah berikut mencetak kedua hasilnya.

```
>pi*2*r*h, %+2*pi*r*h // Ingat tanda % menyatakan hasil perhitungan terakhir
```

50.2654824574
100.530964915

Baris perintah dijalankan sesuai urutan yang ditekan pengguna kembali. Jadi, Anda mendapatkan nilai baru setiap kali Anda menjalankan baris kedua.

```
>x := 1;  
>x := cos(x) // nilai cosinus (x dalam radian)
```

0.540302305868

```
>x := cos(x)
```

0.857553215846

Jika dua jalur dihubungkan dengan "..." kedua jalur akan selalu dijalankan secara bersamaan.

```
>x := 1.5; ...
>x := (x+2/x)/2, x := (x+2/x)/2, x := (x+2/x)/2,
```

1.4166666667
1.41421568627
1.41421356237

Ini juga merupakan cara yang baik untuk menyebarluaskan perintah panjang ke dua baris atau lebih. Anda dapat menekan Ctrl+Return untuk membagi baris menjadi dua pada posisi kursor saat ini, atau Ctrl+Back untuk menggabungkan baris.

Untuk melipat semua multi-garis tekan Ctrl+L. Maka garis-garis berikutnya hanya akan terlihat, jika salah satunya mendapat fokus. Untuk melipat satu multi-baris, mulailah baris pertama dengan "%+".

```
>%+ x=4+5; ...
```

Baris yang dimulai dengan %% tidak akan terlihat sama sekali.

81

Euler mendukung loop di baris perintah, asalkan cocok ke dalam satu baris atau multi-baris. Tentu saja, pembatasan ini tidak berlaku dalam program. Untuk informasi lebih lanjut lihat pendahuluan berikut.

```
>x=1; for i=1 to 5; x := (x+2/x)/2, end; // menghitung akar 2
```

1.5
1.4166666667
1.41421568627
1.41421356237
1.41421356237

Tidak apa-apa menggunakan multi-baris. Pastikan baris diakhiri dengan "...".

```
>x := 1.5; // comments go here before the ...
>repeat xnew:=(x+2/x)/2; until xnew~=x; ...
>    x := xnew; ...
>end; ...
>x,
```

1.41421356237

Struktur bersyarat juga berfungsi.

```
>if E^pi>pi^E; then "Thought so!", endif;
```

Thought so!

Saat Anda menjalankan perintah, kursor dapat berada di posisi mana pun di baris perintah. Anda dapat kembali ke perintah sebelumnya atau melompat ke perintah berikutnya dengan tombol panah. Atau Anda dapat mengklik bagian komentar di atas perintah untuk membuka perintah.

Saat Anda menggerakkan kursor di sepanjang garis, pasangan tanda kurung atau tanda kurung buka dan tutup akan disorot. Juga, perhatikan baris status. Setelah tanda kurung buka dari fungsi `sqrt()`, baris status akan menampilkan teks bantuan untuk fungsi tersebut. Jalankan perintah dengan kunci kembali.

```
>sqrt(sin(10°)/cos(20°))
```

0.429875017772

Untuk melihat bantuan untuk perintah terbaru, buka jendela bantuan dengan F1. Di sana, Anda dapat memasukkan teks untuk dicari. Pada baris kosong, bantuan untuk jendela bantuan akan ditampilkan. Anda dapat menekan escape untuk menghapus garis, atau untuk menutup jendela bantuan.

Anda dapat mengklik dua kali pada perintah apa pun untuk membuka bantuan untuk perintah ini. Coba klik dua kali pada `exp` command di bawah pada baris perintah.

```
>exp(log(2.5))
```

2.5

Anda juga dapat menyalin dan menempel di Euler. Gunakan Ctrl-C dan Ctrl-V untuk ini. Untuk menandai teks, seret mouse atau gunakan shift bersamaan dengan tombol kursor apa pun. Selain itu, Anda dapat menyalin tanda kurung yang disorot.

Euler mengetahui fungsi matematika biasa. Seperti yang Anda lihat di atas, fungsi trigonometri bekerja dalam radian atau derajat. Untuk mengonversi ke derajat, tambahkan simbol derajat (dengan tombol F7) ke nilainya, atau gunakan fungsi rad(x). Fungsi akar kuadrat disebut sqrt di Euler. Tentu saja, $x^{(1/2)}$ juga dimungkinkan.

Untuk menyetel variabel, gunakan "=" atau ":=". Demi kejelasan, pendahuluan ini menggunakan bentuk yang terakhir. Spasi tidak penting. Tapi jarak antar perintah diharapkan.

Beberapa perintah dalam satu baris dipisahkan dengan "," atau ";". Titik koma menekan keluaran perintah. Di akhir baris perintah, "," diasumsikan, jika ";" hilang.

```
>g:=9.81; t:=2.5; 1/2*g*t^2
```

30.65625

EMT uses a programming syntax for expressions. To enter

$$e^2 \cdot \left(\frac{1}{3 + 4 \log(0.6)} + \frac{1}{7} \right)$$

you have to set the correct brackets and use / for fractions. Watch the highlighted brackets for assistance. Note that the Euler constant e is named E in EMT.

```
>E^2*(1/(3+4*log(0.6))+1/7)
```

8.77908249441

To compute a complicate expression like

$$\left(\frac{\frac{1}{7} + \frac{1}{8} + 2}{\frac{1}{3} + \frac{1}{2}} \right)^2 \pi$$

you need to enter it in line form.

```
>((1/7 + 1/8 + 2) / (1/3 + 1/2))^2 * pi
```

23.2671801626

Carefully put brackets around sub-expressions that need to be computed first. EMT assists you by highlighting the expression that the closing bracket finishes. You will also have to enter the name "pi" for the Greek letter pi.

The result of this computation is a floating point number. It is by default printed with about 12 digits accuracy. In the following command line, we also learn how we can refer to the previous result within the same line.

```
>1/3+1/7, fraction %
```

0.47619047619

10/21

An Euler command can be an expression or a primitive command. An expression is made of operators and functions. If necessary, it must contain brackets to force the correct order of execution. In doubt, setting a bracket is a good idea. Note that EMT shows opening and closing brackets while editing the command line.

```
> (cos(pi/4)+1)^3*(sin(pi/4)+1)^2
```

14.4978445072

The numerical operators of Euler include

- + unary or operator plus
- unary or operator minus
- *, /
- . the matrix product
- a^b power for positive a or integer b ($a**b$ works too)
- $n!$ the factorial operator

and many more.

Here are some of the functions you might need. There are many more.

- `sin,cos,tan,atan,asin,acos,rad,deg`
- `log,exp,log10,sqrt,logbase`
- `bin,logbin,logfac,mod,floor,ceil,round,abs,sign`
- `conj,re,im,arg,conj,real,complex`
- `beta,betai,gamma,complexgamma,ellrf,ellf,ellrd,elle`
- `bitand,bitor,bitxor,bitnot`

Some commands have aliases, e.g. `ln` for `log`.

```
>ln(E^2), arctan(tan(0.5))
```

2
0.5

```
>sin(30°)
```

0.5

Make sure to use parentheses (round brackets), whenever there is doubt about the order of execution! The following is not the same as $(2^3)^4$, which is the default for 2^3^4 in EMT (some numerical systems do it the other way).

```
>2^3^4, (2^3)^4, 2^(3^4)
```

2.41785163923e+24
4096
2.41785163923e+24

Real Numbers

The primary data type in Euler is the real number. Reals are represented in IEEE format with about 16 decimal digits of accuracy.

```
>longest 1/3
```

0.3333333333333333

The internal dual representation takes 8 bytes.

```
>printdual(1/3)
```

1.01*2^-2

```
>printhex(1/3)
```

5.555555555554*16^-1

Strings

A string in Euler is defined with "...".

```
>"A string can contain anything."
```

A string can contain anything.

Strings can be concatenated with `|` or with `+`. This also works with numbers, which are converted to strings in that case.

```
>"The area of the circle with radius " + 2 + " cm is " + pi*4 + " cm^2."
```

The area of the circle with radius 2 cm is 12.5663706144 cm².

The print function does also convert a number to a string. It can take a number of digits and a number of places (0 for dense output), and optimally a unit.

```
>"Golden Ratio : " + print((1+sqrt(5))/2,5,0)
```

Golden Ratio : 1.61803

There is a special string `none`, which does not print. It is returned by some functions, when the result does not matter. (It is returned automatically, if the function does not have a return statement.)

```
>none
```

To convert a string to a number simply evaluate it. This works for expressions too (see below).

```
>"1234.5"()
```

1234.5

To define a string vector, use the vector [...] notation.

```
>v := ["affe", "charlie", "bravo"]
```

affe
charlie
bravo

The empty string vector is denoted by [none]. String vectors can be concatenated.

```
>w:=[none]; w|v|v
```

```
affe
charlie
bravo
affe
charlie
bravo
```

Strings can contain Unicode characters. Internally, these strings contain UTF-8 code. To generate such a string, use u"..." and one of the HTML entities.

Unicode strings can be concatenated like other strings.

```
>u"&alpha; = " + 45 + u"&deg;" // pdfLaTeX mungkin gagal menampilkan secara
```

```
= 45°
```

I

In comments, the same entities like , etc. can be used. This may be a quick alternative to Latex. (More details on comments below).

There are some functions to create or analyze unicode strings. The function strtochar() will recognize Unicode strings, and translate them correctly.

```
>v=strtochar(u"&Auml; is a German letter")
```

```
[196, 32, 105, 115, 32, 97, 32, 71, 101, 114, 109, 97, 110,
32, 108, 101, 116, 116, 101, 114]
```

The result is a vector of Unicode numbers. The converse function is chartoutf().

```
>v[1]=strtochar(u"&Uuml;") [1]; chartoutf(v)
```

```
Ü is a German letter
```

The function utf() can translate a string with entities in a variable into a Unicode string.

```
>s="We have &alpha;=&beta; ."; utf(s) // pdfLaTeX mungkin gagal menampilkan
```

```
We have =.
```

It is also possible to use numerical entities.

```
>u"hnliches"
```

Ähnliches

Boolean Values

Boolean values are represented with 1=true or 0=false in Euler. Strings can be compared, just like numbers.

```
>2<1, "apel"<"banana"
```

```
0  
1
```

"and" is the operator "&&" and "or" is the operator "||", as in the C language. (The words "and" and "or" can only be used in conditions for "if".)

```
>2<E && E<3
```

```
1
```

Boolean operators obey the rules of the matrix language.

```
>(1:10)>5, nonzeros(%)
```

```
[0, 0, 0, 0, 0, 1, 1, 1, 1, 1]  
[6, 7, 8, 9, 10]
```

You can use the function nonzeros() to extract specific elements from a vector. In the example, we use the conditional isprime(n).

```
>N=2|3:2:99 // N berisi elemen 2 dan bilangan2 ganjil dari 3 s.d. 99
```

```
[2, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29,  
31, 33, 35, 37, 39, 41, 43, 45, 47, 49, 51, 53, 55, 57,  
59, 61, 63, 65, 67, 69, 71, 73, 75, 77, 79, 81, 83, 85,  
87, 89, 91, 93, 95, 97, 99]
```

```
>N[nonzeros(isprime(N))] //pilih anggota2 N yang prima
```

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47,
53, 59, 61, 67, 71, 73, 79, 83, 89, 97]
```

Output Formats

The default output format of EMT prints 12 digits. To make sure that we see the default, we reset the format.

```
>defformat; pi
```

```
3.14159265359
```

Internally, EMT uses the IEEE standard for double numbers with about 16 decimal digits. To see the full number of digits, use the command "longestformat", or we use the operator "longest" to display the result in the longest format.

```
>longest pi
```

```
3.141592653589793
```

Here is the internal hexadecimal representation of a double number.

```
>printhex(pi)
```

```
3.243F6A8885A30*16^0
```

The output format can be changed permanently with a format command.

```
>format(12,5); 1/3, pi, sin(1)
```

```
0.33333
3.14159
0.84147
```

The default is format(12).

```
>format(12); 1/3
```

0.333333333333

Functions like "shortestformat", "shortformat", "longformat" work for vectors in the following way.

```
>shortestformat; random(3,8)
```

0.66	0.2	0.89	0.28	0.53	0.31	0.44	0.3
0.28	0.88	0.27	0.7	0.22	0.45	0.31	0.91
0.19	0.46	0.095	0.6	0.43	0.73	0.47	0.32

The default format for scalars is format(12). But this can be changed.

```
>setscalarformat(5); pi
```

3.1416

The function "longestformat" set the scalar format too.

```
>longestformat; pi
```

3.141592653589793

For reference, here is a list of the most important output formats.

```
shortestformat shortformat longformat, longestformat
format(length,digits) goodformat(length)
fracformat(length)
defformat
```

The internal accuracy of EMT is about 16 decimal places, which is the IEEE standard. Numbers are stored in this internal format.

But the output format of EMT can be set in a flexible way.

```
>longestformat; pi,
```

3.141592653589793

```
>format(10,5); pi
```

3.14159

The default is deformat().

```
>deformat; // default
```

There are short operators which print only one value. The operator "longest" will print all valid digits of a number.

```
>longest pi^2/2
```

4.934802200544679

There is also a short operator for printing a result in fractional format. We have already used it above.

```
>fraction 1+1/2+1/3+1/4
```

25/12

Since the internal format uses a binary way to store numbers, the value 0.1 will not be represented exactly. The error adds up a bit, as you see in the following computation.

```
>longest 0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1-1
```

-1.110223024625157e-16

But with the default "longformat" you will not notice this. For convenience, the output of very small numbers is 0.

```
>0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1-1
```

0

Expressions

Strings or names can be used to store mathematical expressions, which can be evaluated by EMT. For this, use parentheses after the expression. If you intend to use a string as an expression, use the convention to name it "fx" or "fxy" etc. Expressions take precedence over functions.

Global variables can be used in the evaluation.

```
>r:=2; fx:="pi*r^2"; longest fx()
```

12.56637061435917

Parameters are assigned to x, y, and z in that order. Additional parameters can be added using assigned parameters.

```
>fx:="a*sin(x)^2"; fx(5,a=-1)
```

-0.919535764538

Note that expression will always use global variables, even if there is a variable in a function with the same name. (Otherwise the evaluation of expressions in functions could have very confusing results for the user that called the function.)

```
>at:=4; function f(expr,x,at) := expr(x); ...
>f("at*x^2",3,5) // computes 4*3^2 not 5*3^2
```

36

If you want to use another value for "at" than the global value you need to add "at=value".

```
>at:=4; function f(expr,x,a) := expr(x,at=a); ...
>f("at*x^2",3,5)
```

45

For reference, we remark that call collections (discussed elsewhere) can contain expressions. So we can make the above example as follows.

```
>at:=4; function f(expr,x) := expr(x); ...
>f({{"at*x^2",at=5}},3)
```

45

Expressions in x are often used just like functions.

Note that defining a function with the same name like a global symbolic expression deletes this variable to avoid confusion between symbolic expressions and functions.

```
>f &= 5*x;
>function f(x) := 6*x;
>f(2)
```

12

By way of convention, symbolic or numerical expressions should be named fx , fxy etc. This naming scheme should not be used for functions.

```
>fx &= diff(x^x, x); $&fx
```

A special form of an expression allows any variable as an unnamed parameter to the evaluation of the expression, not just " x ", " y " etc. For this, start the expression with "@(variables) ...".

```
>"@(a,b) a^2+b^2", %(4,5)
```

@(a,b) a^2+b^2

41

This allows to manipulate expressions in other variables for functions of EMT which need an expression in " x ".

The most elementary way to define a simple function is to store its formula in a symbolic or numerical expression. If the main variable is x , the expression can be evaluated just like a function.

As you see in the following example, global variables are visible during the evaluation.

```
>fx &= x^3-a*x; ...
>a=1.2; fx(0.5)
```

-0.475

All other variables in the expression can be specified in the evaluation using an assigned parameter.

```
>fx(0.5,a=1.1)
```

-0.425

An expression needs not be symbolic. This is necessary, if the expression contains functions, which are only known in the numerical kernel, not in Maxima.

Symbolic Mathematics

EMT does symbolic math with the help of Maxima. For details, start with the following tutorial, or browse the reference for Maxima. Experts in Maxima should note that there are differences in the syntax between the original syntax of Maxima and the default syntax of symbolic expressions in EMT.

Symbolic math is integrated seamlessly into Euler with &. Any expression starting with & is a symbolic expression. It is evaluated and printed by Maxima.

First of all, Maxima has an "infinite" arithmetic which can handle very large numbers.

```
>$&44!
```

This way, you can compute large results exactly. Let us compute

$$C(44, 10) = \frac{44!}{34! \cdot 10!}$$

```
>$& 44!/ (34!*10!) // nilai C(44,10)
```

Of course, Maxima has a more efficient function for this (as does the numerical part of EMT).

```
>$binomial(44,10) //menghitung C(44,10) menggunakan fungsi binomial()
```

To learn more about a specific function double click on it. E.g., try double clicking on "&binomial" in the previous command line. This opens the documentation of Maxima as provided by the authors of that program.

You will learn that the following works too.

$$C(x, 3) = \frac{x!}{(x - 3)!3!} = \frac{(x - 2)(x - 1)x}{6}$$

```
>$binomial(x, 3) // C(x, 3)
```

If you want to replace x with any specific value use "with".

```
>${&}binomial(x, 3) with x=10 // substitusi x=10 ke C(x, 3)
```

That way you can use a solution of an equation in another equation.

Symbolic expressions are printed by Maxima in 2D form. The reason for this is a special symbolic flag in the string.

As you will have seen in previous and following examples, if you have LaTeX installed, you can print a symbolic expression with Latex. If not, the following command will issue an error message.

To print a symbolic expression with LaTeX, use \$ in front of & (or you may omit &) before the command. Do not run the Maxima commands with \$, if you don't have LaTeX installed.

```
>$(3+x)/(x^2+1)
```

Symbolic expressions are parsed by Euler. If you need a complex syntax in one expression, you can enclose the expression in "...". To use more than a simple expression is possible, but strongly discouraged.

```
>&"v := 5; v^2"
```

25

For completeness, we remark that symbolic expressions can be used in programs, but need to be enclosed in quotes. Moreover, it is much more effective to call Maxima at compile time if possible.

```
>${&}expand((1+x)^4), ${&}factor(diff(% ,x)) // diff: turunan, factor: faktor
```

Again, % refers to the previous result.

To make things easier we save the solution to a symbolic variable. Symbolic variables are defined with "&=".

```
>fx &= (x+1) / (x^4+1); $&fx
```

Symbolic expressions can be used in other symbolic expressions.

```
>$&factor(diff(fx,x))
```

A direct input of Maxima commands is available too. Start the command line with "::". The syntax of Maxima is adapted to the syntax of EMT (called the "compatibility mode").

```
>&factor(20!)
```

2432902008176640000

```
>::: factor(10!)
```

$$\begin{matrix} 8 & 4 & 2 \\ 2 & 3 & 5 & 7 \end{matrix}$$

```
>::: factor(20!)
```

$$\begin{matrix} 18 & 8 & 4 & 2 \\ 2 & 3 & 5 & 7 & 11 & 13 & 17 & 19 \end{matrix}$$

If you are an expert in Maxima, you may wish to use the original syntax of Maxima. You can do this with ":::".

```
>::: av:g$ av^2;
```

$$\begin{matrix} 2 \\ g \end{matrix}$$

```
>fx &= x^3*exp(x), $fx
```

$$\frac{x^3}{e}$$

Such variables can be used in other symbolic expressions. Note, that in the following command the right hand side of `&=` is evaluated before the assignment to `Fx`.

```
>&(fx with x=5), $%, &float(%)
```

$$\frac{125}{e^5}$$

$$18551.64488782208$$

```
>fx(5)
```

$$18551.6448878$$

For the evaluation of an expression with specific values of the variables, you can use the "with" operator.

The following command line also demonstrates that Maxima can evaluate an expression numerically with `float()`.

```
>&(fx with x=10)-(fx with x=5), &float(%)
```

$$1000 \frac{e^{10}}{e} - \frac{125}{e^5}$$

$$2.20079141499189e+7$$

```
>$factor(diff(fx,x,2))
```

To get the Latex code for an expression, you can use the tex command.

```
>tex(fx)
```

$x^3 \backslash, e^{x}$

Symbolic expressions can be evaluated just like numerical expressions.

```
>fx(0.5)
```

0.206090158838

In symbolic expressions, this does not work, since Maxima does not support it. Instead, use the "with" syntax (a nicer form of the at(...) command of Maxima).

```
>$&fx with x=1/2
```

The assignment can also be symbolic.

```
>$&fx with x=1+t
```

The command solve solves symbolic expressions for a variable in Maxima. The result is a vector of solutions.

```
>$&solve(x^2+x=4,x)
```

Compare with the numerical "solve" command in Euler, which needs a start value, and optionally a target value.

```
>solve("x^2+x",1,y=4)
```

1.56155281281

The numerical values of the symbolic solution can be computed by evaluation of the symbolic result. Euler will read over the assignments x= etc. If you do not need the numerical results for further computations you can also let Maxima find the numerical values.

```
>sol &= solve(x^2+2*x=4,x); $&sol, sol(), $&float(sol)
```

$[-3.23607, 1.23607]$

To get a specific symbolic solution, one can use "with" and an index.

```
>$&solve(x^2+x=1,x), x2 &= x with %[2]; $&x2
```

To solve a system of equations, use a vector of equations. The result is a vector of solutions.

```
>sol &= solve([x+y=3,x^2+y^2=5],[x,y]); $&sol, $&x*y with sol[1]
```

Symbolic expressions can have flags, which indicate a special treatment in Maxima. Some flags can be used as commands too, others can't. Flags are appended with "`|`" (a nicer form of "`ev(...,flags)"`)

```
>$& diff((x^3-1)/(x+1),x) //turunan bentuk pecahan
```

$$\frac{1 - x^3}{(1 + x)^2} + \frac{3 x^2}{1 + x}$$

```
>$& diff((x^3-1)/(x+1),x) | ratsimp //menyederhanakan pecahan
```

$$\frac{1 + 3 x^2 + 2 x^3}{1 + 2 x + x^2}$$

```
>$&factor(%)
```

Functions

In EMT, functions are programs defined with the command "function". It can be a one-line function or multiline function.

A one-line function can be numerical or symbolic. A numerical one-line function is defined by "`:=`".

```
>function f(x) := x*sqrt(x^2+1)
```

For an overview, we show all possible definitions for one-line functions. A function can be evaluated just like any built-in Euler function.

```
>f(2)
```

4.472135955

This function will work for vectors too, obeying the matrix language of Euler, since the expressions used in the function are vectorized.

```
>f(0:0.1:1)
```

[0, 0.100499, 0.203961, 0.313209, 0.430813, 0.559017, 0.699714,
0.854459, 1.0245, 1.21083, 1.41421]

Functions can be plotted. Instead of expressions, we need only provide the function name. In contrast to symbolic or numerical expressions, the function name must be provided in a string.

```
>solve("f", 1, y=1)
```

0.786151377757

By default, if you need to overwrite a built-in function, you must add the keyword "overwrite". Overwriting built-in functions is dangerous and can cause problems for other functions depending on them.

You can still call the built-in function as "...", if it is function in the Euler core.

```
>function overwrite sin (x) := _sin(x°) // redefine sine in degrees
>sin(45)
```

0.707106781187

We better remove this redefinition of sin.

```
>forget sin; sin(pi/4)
```

0.707106781187

Default Parameters

Numerical function can have default parameters.

```
>function f(x, a=1) := a*x^2
```

Omitting this parameter uses the default value.

```
>f(4)
```

16

Setting it overwrites the default value.

```
>f(4, 5)
```

80

An assigned parameter overwrite it too. This is used by many Euler functions like plot2d, plot3d.

```
>f(4, a=1)
```

16

If a variable is not a parameter, it must be global. One-line functions can see global variables.

```
>function f(x) := a*x^2
>a=6; f(2)
```

24

But an assigned parameter overrides the global value.

If the argument is not in the list of pre-defined parameters, it must be declared with ":="!

```
>f(2,a:=5)
```

20

Symbolic functions are defined with "&=". They are defined in Euler and Maxima, and work in both worlds. The defining expression is run through Maxima before the definition.

```
>function g(x) &= x^3-x*exp(-x); $&g(x)
```

Symbolic functions can be used in symbolic expressions.

```
>$&diff(g(x),x), $&% with x=4/3
```

They can also be used in numerical expressions. Of course, this will only work if EMT can interpret everything inside the function.

```
>g(5+g(1))
```

178.635099908

They can be used to define other symbolic functions or expressions.

```
>function G(x) &= factor(integrate(g(x),x)); $&G(c) // integrate: menginteg
>solve(&g(x),0.5)
```

0.703467422498

The following works too, since Euler uses the symbolic expression in the function *g*, if it does not find a symbolic variable *g*, and if there is a symbolic function *g*.

```
>solve(&g,0.5)
```

0.703467422498

```
>function P(x,n) &= (2*x-1)^n; $&P(x,n)
>function Q(x,n) &= (x+2)^n; $&Q(x,n)
>$&P(x,4), $&expand(%)
>P(3,4)
```

625

```
>$&P(x,4)+Q(x,3), $&expand(%)
```

$$Q(x,3) + P(x,4)$$

$$Q(x,3) + P(x,4)$$

```
>$&P(x,4)-Q(x,3), $&expand(%), $&factor(%)
```

$$-Q(x,3) + P(x,4)$$

$$-Q(x,3) + P(x,4)$$

$$-Q(x,3) + P(x,4)$$

```
>$&P(x,4)*Q(x,3), $&expand(%), $&factor(%)
```

$$Q(x,3) P(x,4)$$

$$Q(x,3) P(x,4)$$

$$Q(x,3) P(x,4)$$

```
>$&P(x,4)/Q(x,1), $&expand(%), $&factor(%)
```

$$\frac{P(x,4)}{Q(x,1)}$$

$$\frac{P(x, 4)}{Q(x, 1)}$$

$$\frac{P(x, 4)}{Q(x, 1)}$$

```
>function f(x) &= x^3-x; $&f(x)
```

With `&=` the function is symbolic, and can be used in other symbolic expressions.

```
>$&integrate(f(x), x)
```

With `:=` the function is numerical. A good example is a definite integral like

$$f(x) = \int_1^x t^t dt,$$

which can not be evaluated symbolically.

If we redefine the function with the keyword "map" it can be used for vectors x. Internally, the function is called for all values of x once, and the results are stored in a vector.

```
>function map f(x) := integrate("x^x", 1, x)
>f(0:0.5:2)
```

`[-0.783431, -0.410816, 0, 0.676863, 2.05045]`

Functions can have default values for parameters.

```
>function mylog (x, base=10) := ln(x)/ln(base);
```

Now the function can be called with or without a parameter "base".

```
>mylog(100), mylog(2^6.7, 2)
```

2
6.7

Moreover, it is possible to use assigned parameters.

```
>mylog(E^2,base=E)
```

2

Often, we want to use functions for vectors at one place, and for individual elements at other places. This is possible with vector parameters.

```
>function f([a,b]) &= a^2+b^2-a*b+b; $&f(a,b), $&f(x,y)
```

Such a symbolic function can be used for symbolic variables.

But the function can also be used for a numerical vector.

```
>v=[3,4]; f(v)
```

17

There are also purely symbolic functions, which cannot be used numerically.

```
>function lapl(expr,x,y) &&= diff(expr,x,2)+diff(expr,y,2)//turunan parsial
```

diff(expr, y, 2) + diff(expr, x, 2)

```
>$&realpart((x+I*y)^4), $&lapl(% ,x,y)
```

But of course, they can be used in symbolic expressions or in the definition of symbolic functions.

```
>function f(x,y) &= factor(lapl((x+y^2)^5,x,y)); $&f(x,y)
```

To summarize

- &= defines symbolic functions,
- := defines numerical functions,
- &&= defines purely symbolic functions.

Solving Expressions

Expressions can be solved numerically and symbolically.

To solve a simple expression of one variable, we can use the `solve()` function. It needs a start value to start the search. Internally, `solve()` uses the secant method.

```
>solve("x^2-2", 1)
```

1.41421356237

This works for symbolic expression too. Take the following function.

```
>$&solve(x^2=2, x)
```

$$\left[x = -\sqrt{2}, x = \sqrt{2} \right]$$

```
>$&solve(x^2-2, x)
```

$$\left[x = -\sqrt{2}, x = \sqrt{2} \right]$$

```
>$&solve(a*x^2+b*x+c=0, x)
```

$$\left[x = \frac{-b - \sqrt{b^2 - 4ac}}{2a}, x = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \right]$$

```
>$&solve( [a*x+b*y=c, d*x+e*y=f] , [x, y] )
```

$$\left[\left[x = \frac{-ce + bf}{bd - ae}, y = \frac{cd - af}{bd - ae} \right] \right]$$

```
>px &= 4*x^8+x^7-x^4-x; $&px
```

$$-x - x^4 + x^7 + 4x^8$$

Now we search the point, where the polynomial is 2. In `solve()`, the default target value `y=0` can be changed with an assigned variable.

We use `y=2` and check by evaluating the polynomial at the previous result.

```
>solve(px, 1, y=2), px(%)
```

```
0.966715594851  
2
```

Solving a symbolic expression in symbolic form returns a list of solutions. We use the symbolic solver `solve()` provided by Maxima.

```
>sol &= solve(x^2-x-1, x); $&sol
```

The easiest way to get the numerical values is to evaluate the solution numerically just like an expression.

```
>longest sol()
```

```
-0.6180339887498949 1.618033988749895
```

To use the solutions symbolically in other expressions, the easiest way is "with".

```
>$&x^2 with sol[1], $&expand(x^2-x-1 with sol[2])
```

Solving systems of equations symbolically can be done with vectors of equations and the symbolic solver `solve()`. The answer is a list of lists of equations.

```
>$&solve([x+y=2, x^3+2*y+x=4], [x, y])
```

The function `f()` can see global variables. But often we want to use local parameters.

$$a^x - x^a = 0.1$$

with `a=3`.

```
>function f(x, a) := x^a-a^x;
```

One way to pass the additional parameter to f() is to use a list with the function name and the parameters (the other way are semicolon parameters).

```
>solve({{"f", 3}}, 2, y=0.1)
```

2.54116291558

This does also work with expressions. But then, a named list element has to be used. (More on lists in the tutorial about the syntax of EMT).

```
>solve({{"x^a-a^x", a=3}}, 2, y=0.1)
```

2.54116291558

Menyelesaikan Pertidaksamaan

Untuk menyelesaikan pertidaksamaan, EMT tidak akan dapat melakukannya, melainkan dengan bantuan Maxima, artinya secara eksak (simbolik). Perintah Maxima yang digunakan adalah fourier_elim(), yang harus dipanggil dengan perintah "load(fourier_elim)" terlebih dahulu.

```
>&load(fourier_elim)
```

C:/Program Files/Euler x64/maxima/share/maxima/5.35.1/share/f\\ourier_elim/fourier_elim.lisp

```
>$&fourier_elim([x^2 - 1 > 0], [x]) // x^2-1 > 0
```

$$\text{fourier_elim} ([x^2 - 1 > 0], [x])$$

```
>$&fourier_elim([x^2 - 1 < 0], [x]) // x^2-1 < 0
```

$$\text{fourier_elim} ([-1 + x^2 < 0], [x])$$

```
> $&fourier_elim([x^2 - 1 # 0], [x]) // x^-1 <> 0
```

$$\text{fourier_elim}([-1 + x^2 \neq 0], [x])$$

```
> $&fourier_elim([x # 6], [x])
```

$$\text{fourier_elim}(x \neq 6, [x])$$

```
> $&fourier_elim([x < 1, x > 1], [x]) // tidak memiliki penyelesaian
```

$$\text{fourier_elim}(x < 1, x > 1), [x])$$

```
> $&fourier_elim([minf < x, x < inf], [x]) // solusinya R
```

$$\text{fourier_elim}(-\infty < x, x < \infty), [x])$$

```
> $&fourier_elim([x^3 - 1 > 0], [x])
```

$$\text{fourier_elim}([-1 + x^3 > 0], [x])$$

```
> $&fourier_elim([cos(x) < 1/2], [x]) // ??? gagal
```

$$\text{fourier_elim}\left(\left[\cos x < \frac{1}{2}\right], [x]\right)$$

```
> $&fourier_elim([y-x < 5, x - y < 7, 10 < y], [x, y]) // sistem pertidaksamaan
```

$$\text{fourier_elim}(-x + y < 5, x - y < 7, 10 < y), [x, y])$$

```
>${&fourier_elim([y-x < 5, x - y < 7, 10 < y], [y, x])}
```

$$\text{fourier_elim}([-x + y < 5, x - y < 7, 10 < y], [y, x])$$

```
>${&fourier_elim((x + y < 5) and (x - y > 8), [x, y])}
```

$$\text{fourier_elim}(x + y < 5 \wedge x - y > 8, [x, y])$$

```
>${&fourier_elim(((x + y < 5) and x < 1) or (x - y > 8), [x, y])}
```

$$\text{fourier_elim}(x + y < 5 \wedge x < 1 \vee x - y > 8, [x, y])$$

```
>&fourier_elim([max(x, y) > 6, x ≠ 8, abs(y-1) > 12], [x, y])
```

$$\begin{aligned} &[6 < x, x < 8, y < -11] \text{ or } [8 < x, y < -11] \\ &\text{or } [x < 8, 13 < y] \text{ or } [x = y, 13 < y] \text{ or } [8 < x, x < y, 13 < y] \\ &\text{or } [y < x, 13 < y] \end{aligned}$$

```
>${&fourier_elim([(x+6)/(x-9) <= 6], [x])}
```

$$\text{fourier_elim}\left(\left[\frac{6+x}{-9+x} \leq 6\right], [x]\right)$$

The Matrix Language

The documentation of the EMT core contains a detailed discussion on the matrix language of Euler.

Vectors and matrices are entered with square brackets, elements separated by commas, rows separated by semicolons.

```
>A=[1,2;3,4]
```

1
3

2
4

The matrix product is denoted by a dot.

```
>b=[3;4]
```

```
3  
4
```

```
>b' // transpose b
```

```
[3, 4]
```

```
>inv(A) //inverse A
```

```
-2 1  
1.5 -0.5
```

```
>A.b //perkalian matriks
```

```
11  
25
```

```
>A.inv(A)
```

```
1 0  
0 1
```

The main point of a matrix language is that all functions and operators work element for element.

```
>A.A
```

```
7 10  
15 22
```

```
>A^2 //perpangkatan elemen2 A
```

```
1 4  
9 16
```

>A.A.A

$$\begin{array}{r} 37 \\ 81 \end{array} \quad \begin{array}{r} 54 \\ 118 \end{array}$$

>power(A, 3) //perpangkatan matriks

$$\begin{array}{r} 37 \\ 81 \end{array} \quad \begin{array}{r} 54 \\ 118 \end{array}$$

>A/A //pembagian elemen-elemen matriks yang seletak

$$\begin{array}{r} 1 \\ 1 \end{array} \quad \begin{array}{r} 1 \\ 1 \end{array}$$

>A/b //pembagian elemen2 A oleh elemen2 b kolom demi kolom (karena b vektor)

$$\begin{array}{r} 0.333333 \\ 0.75 \end{array} \quad \begin{array}{r} 0.666667 \\ 1 \end{array}$$

>A\b // hasil kali invers A dan b, A^{-1}b

$$\begin{array}{r} -2 \\ 2.5 \end{array}$$

>inv(A).b

$$\begin{array}{r} -2 \\ 2.5 \end{array}$$

>A\A //A^{-1}A

$$\begin{array}{r} 1 \\ 0 \end{array} \quad \begin{array}{r} 0 \\ 1 \end{array}$$

```
>inv(A) .A
```

1	0
0	1

```
>A*A // perkalian elemen-elemen matriks seletak
```

1	4
9	16

This is not the matrix product, but a multiplication element by element. The same works for vectors.

```
>b^2 // perpangkatan elemen-elemen matriks/vektor
```

9
16

If one of the operands is a vector or a scalar it is expanded in the natural way.

```
>2*A
```

2	4
6	8

E.g., if the operand is a column vector its elements are applied to all rows of A.

```
>[1, 2]*A
```

1	4
3	8

If it is a row vector it is applied to all columns of A.

```
>A*[2, 3]
```

2	6
6	12

One can imagine this multiplication as if the row vector v had been duplicated to form a matrix of the same size as A.

```
>dup([1,2],2) // dup: menduplikasi/menggandakan vektor [1,2] sebanyak 2 kali
```

1	2
1	2

```
>A*dup([1,2],2)
```

1	4
3	8

This does also apply for two vectors where one is a row vector and the other is a column vector. We compute $i \cdot j$ for i,j from 1 to 5. The trick is to multiply 1:5 with its transpose. The matrix language of Euler automatically generates a table of values.

```
>(1:5)*(1:5)' // hasilkali elemen-elemen vektor baris dan vektor kolom
```

1	2	3	4	5
2	4	6	8	10
3	6	9	12	15
4	8	12	16	20
5	10	15	20	25

Again, remember that this is not the matrix product!

```
>(1:5).(1:5)' // hasilkali vektor baris dan vektor kolom
```

55

```
>sum((1:5)*(1:5)) // sama hasilnya
```

55

Even operators like `<` or `==` work in the same way.

```
>(1:10)<6 // menguji elemen-elemen yang kurang dari 6
```

[1, 1, 1, 1, 1, 0, 0, 0, 0]

E.g., we can count the number of elements satisfying a certain condition with the function `sum()`.

```
>sum((1:10)<6) // banyak elemen yang kurang dari 6
```

5

Euler has comparison operators, like "`==`", which checks for equality.

We get a vector of 0 and 1, where 1 stands for true.

```
>t=(1:10)^2; t==25 //menguji elemen2 t yang sama dengan 25 (hanya ada 1)
```

[0, 0, 0, 0, 1, 0, 0, 0, 0]

From such a vector, "nonzeros" selects the non-zero elements.

In this case, we get the indices of all elements greater than 50.

```
>nonzeros(t>50) //indeks elemen2 t yang lebih besar daripada 50
```

[8, 9, 10]

Of course, we can use this vector of indices to get the corresponding values in t.

```
>t[nonzeros(t>50)] //elemen2 t yang lebih besar daripada 50
```

[64, 81, 100]

As an example, let us find all squares of the numbers 1 to 1000, which are 5 modulo 11 and 3 modulo 13.

```
>t=1:1000; nonzeros(mod(t^2,11)==5 && mod(t^2,13)==3)
```

[4, 48, 95, 139, 147, 191, 238, 282, 290, 334, 381, 425, 433, 477, 524, 568, 576, 620, 667, 711, 719, 763, 810, 854, 862, 906, 953, 997]

EMT is not completely effective for integer computations. It uses double precision floating point internally. However, it is often very useful.

We can check for primality. Let us find out, how many squares plus 1 are primes.

```
>t=1:1000; length(nonzeros(isprime(t^2+1)))
```

The function `nonzeros()` works only for vectors. For matrices, there is `mnonzeros()`.

```
>seed(2); A=random(3, 4)
```

0.765761	0.401188	0.406347	0.267829
0.13673	0.390567	0.495975	0.952814
0.548138	0.006085	0.444255	0.539246

It returns the indices of the elements, which are not zeros.

```
>k=mnonzeros(A<0.4) //indeks elemen2 A yang kurang dari 0,4
```

1	4
2	1
2	2
3	2

These indices can be used to set the elements to some value.

```
>mset(A, k, 0) //mengganti elemen2 suatu matriks pada indeks tertentu
```

0.765761	0.401188	0.406347	0
0	0	0.495975	0.952814
0.548138	0	0.444255	0.539246

The function `mset()` can also set the elements at the indices to the entries of some other matrix.

```
>mset(A, k, -random(size(A)))
```

0.765761	0.401188	0.406347	-0.126917
-0.122404	-0.691673	0.495975	0.952814
0.548138	-0.483902	0.444255	0.539246

And it is possible to get the elements in a vector.

```
>mget(A, k)
```

```
[0.267829, 0.13673, 0.390567, 0.006085]
```

Another useful function is `extrema`, which returns the minimal and maximal values in each row of the matrix and their positions.

```
>ex=extrema(A)
```

0.267829	4	0.765761	1
0.13673	1	0.952814	4
0.006085	2	0.548138	1

We can use this to extract the maximal values in each row.

```
>ex[,3]'
```

[0.765761, 0.952814, 0.548138]

This, of course, is the same as the function max().

```
>max(A)'
```

[0.765761, 0.952814, 0.548138]

But with mget(), we can extract the indices and use this information to extract the elements at the same positions from another matrix.

```
>j=(1:rows(A))' | ex[,4], mget(-A, j)
```

1	1
2	4
3	1
[-0.765761, -0.952814, -0.548138]	

Other Matrix Functions (Building Matrix)

To build a matrix, we can stack one matrix on top of another. If both do not have the same number of columns, the shorter one will be filled with 0.

```
>v=1:3; v_v
```

1	2	3
1	2	3

Likewise, we can attach a matrix to another side by side, if both have the same number of rows.

```
>A=random(3,4); A|v'
```

0.032444	0.0534171	0.595713	0.564454	1
0.83916	0.175552	0.396988	0.83514	2
0.0257573	0.658585	0.629832	0.770895	3

If they do not have the same number of rows the shorter matrix is filled with 0.

There is an exception to this rule. A real number attached to a matrix will be used as a column filled with that real number.

```
>A|1
```

0.032444	0.0534171	0.595713	0.564454	1
0.83916	0.175552	0.396988	0.83514	1
0.0257573	0.658585	0.629832	0.770895	1

It is possible to make a matrix of row and column vectors.

```
>[v;v]
```

1	2	3
1	2	3

```
>[v',v']
```

1	1
2	2
3	3

The main purpose of this is to interpret a vector of expressions for column vectors.

```
>"[x,x^2]"(v')
```

1	1
2	4
3	9

To get the size of A, we can use the following functions.

```
>C=zeros(2,4); rows(C), cols(C), size(C), length(C)
```

```
2
4
[2, 4]
4
```

For vectors, there is `length()`.

```
>length(2:10)
```

```
9
```

There are many other functions, which generate matrices.

```
>ones(2,2)
```

```
1           1
1           1
```

This can also be used with one parameter. To get a vector with another number than 1, use the following.

```
>ones(5)*6
```

```
[6, 6, 6, 6, 6]
```

Also a matrix of random numbers can be generated with `random` (uniform distribution) or `normal` (Gauß distribution).

```
>random(2,2)
```

```
0.66566      0.831835
0.977        0.544258
```

Here is another useful function, which restructures the elements of a matrix into another matrix.

```
>redim(1:9,3,3) // menyusun elemen2 1, 2, 3, ..., 9 ke bentuk matriks 3x3
```

```
1           2           3
4           5           6
7           8           9
```

With the following function, we can use this and the `dup` function to write a `rep()` function, which repeats a vector `n` times.

```
>function rep(v,n) := redim(dup(v,n),1,n*cols(v))
```

Let us test.

```
>rep(1:3,5)
```

```
[1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3]
```

The function `multdup()` duplicates elements of a vector.

```
>multdup(1:3,5), multdup(1:3,[2,3,2])
```

```
[1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 3, 3]
[1, 1, 2, 2, 3, 3]
```

The functions `flipx()` and `flipy()` revert the order of the rows or columns of a matrix. I.e., the function `flipx()` flips horizontally.

```
>flipx(1:5) // membalik elemen2 vektor baris
```

```
[5, 4, 3, 2, 1]
```

For rotations, Euler has `rotleft()` and `rotright()`.

```
>rotleft(1:5) // memutar elemen2 vektor baris
```

```
[2, 3, 4, 5, 1]
```

A special function is `drop(v,i)`, which removes the elements with the indices in `i` from the vector `v`.

```
>drop(10:20,3)
```

```
[10, 11, 13, 14, 15, 16, 17, 18, 19, 20]
```

Note that the vector `i` in `drop(v,i)` refers to indices of elements in `v`, not the values of the elements. If you want to remove elements, you need to find the elements first. The function `indexof(v,x)` can be used to find elements `x` in a sorted vector `v`.

```
>v=primes(50), i=indexof(v,10:20), drop(v,i)
```

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47]
[0, 5, 0, 6, 0, 0, 0, 7, 0, 8, 0]
[2, 3, 5, 7, 23, 29, 31, 37, 41, 43, 47]
```

As you see, it does not harm to include indices out of range (like 0), double indices, or unsorted indices.

```
>drop(1:10,shuffle([0,0,5,5,7,12,12]))
```

```
[1, 2, 3, 4, 6, 8, 9, 10]
```

There are some special functions to set diagonals or to generate a diagonal matrix.

We start with the identity matrix.

```
>A=id(5) // matriks identitas 5x5
```

1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1

Then we set the lower diagonal (-1) to 1:4.

```
>setdiag(A,-1,1:4) //mengganti diagonal di bawah diagonal utama
```

1	0	0	0	0
1	1	0	0	0
0	2	1	0	0
0	0	3	1	0
0	0	0	4	1

Note that we did not change the matrix A. We get a new matrix as result of setdiag().
Here is a function, which returns a tri-diagonal matrix.

```
>function tridiag (n,a,b,c) := setdiag(setdiag(b*id(n),1,c),-1,a); ...
>tridiag(5,1,2,3)
```

2	3	0	0	0
1	2	3	0	0
0	1	2	3	0
0	0	1	2	3
0	0	0	1	2

The diagonal of a matrix can also be extracted from the matrix. To demonstrate this, we restructure the vector 1:9 to a 3x3 matrix.

```
>A=redim(1:9, 3, 3)
```

1	2	3
4	5	6
7	8	9

Now we can extract the diagonal.

```
>d=getdiag(A, 0)
```

[1, 5, 9]

E.g. We can divide the matrix by its diagonal. The matrix language takes care that the column vector d is applied to the matrix row by row.

```
>fraction A/d'
```

1	2	3
4/5	1	6/5
7/9	8/9	1

Vectorization

Almost all functions in Euler work for matrix and vector input too, whenever this makes sense.

E.g., the sqrt() function computes the square root of all elements of the vector or matrix.

```
>sqrt(1:3)
```

[1, 1.41421, 1.73205]

So you can easily create a table of values. This is one way to plot a function (the alternative uses an expression).

```
>x=1:0.01:5; y=log(x)/x^2; // terlalu panjang untuk ditampilkan
```

With this and the colon operator a:delta:b, vectors of values of functions can be generated easily.

In the following example, we generate a vector of values $t[i]$ with spacing 0.1 from -1 to 1. Then we generate a vector of values of the function

$$s = t^3 - t$$

```
>t=-1:0.1:1; s=t^3-t
```

```
[0, 0.171, 0.288, 0.357, 0.384, 0.375, 0.336, 0.273, 0.192,
0.099, 0, -0.099, -0.192, -0.273, -0.336, -0.375, -0.384,
-0.357, -0.288, -0.171, 0]
```

EMT expands operators for scalars, vectors, and matrices in the obvious way.

E.g., a column vector times a row vector expands to matrix, if an operator is applied. In the following, v' is the transposed vector (a column vector).

```
>shortest (1:5)*(1:5)'
```

1	2	3	4	5
2	4	6	8	10
3	6	9	12	15
4	8	12	16	20
5	10	15	20	25

Note, that this is quite different from the matrix product. The matrix product is denoted with a dot "." in EMT.

```
>(1:5).(1:5)'
```

55

By default, row vectors are printed in a compact format.

```
>[1,2,3,4]
```

```
[1, 2, 3, 4]
```

For matrices the special operator \cdot denotes matrix multiplication, and A' denotes transposing. A 1×1 matrix can be used just like a real number.

```
>v:=[1,2]; v.v', %^2
```

5

25

To transpose a matrix we use the apostrophe.

```
>v=1:4; v'
```

1

2

3

4

So we can compute matrix A times vector b.

```
>A=[1,2,3,4;5,6,7,8]; A.v'
```

30

70

Note that v is still a row vector. So $v' \cdot v$ is different from $v \cdot v'$.

```
>v' . v
```

1

2

3

4

2

4

6

8

3

6

9

12

4

8

12

16

$v \cdot v'$ computes the norm of v squared for row vectors v. The result is a 1×1 vector, which works just like a real number.

```
>v.v'
```

30

There is also the function norm (along with many other function of Linear Algebra).

```
>norm(v)^2
```

30

Operators and functions obey the matrix language of Euler.

Here is a summary of the rules.

- A function applied to a vector or matrix is applied to each element.
- An operator operating on two matrices of same size is applied pairwise to the elements of the matrices.
- If the two matrices have different dimensions, both are expanded in a sensible way, so that they have the same size.

E.g., a scalar value times a vector multiplies the value with each element of the vector. Or a matrix times a vector (with *, not .) expands the vector to the size of the matrix by duplicating it.

The following is a simple case with the operator ^.

```
>[1,2,3]^2
```

[1, 4, 9]

Here is a more complicated case. A row vector times a column vector expands both by duplicating.

```
>v:=[1,2,3]; v*v'
```

1	2	3
2	4	6
3	6	9

Note that the scalar product uses the matrix product, not the *!

```
>v.v'
```

14

There are numerous functions for matrices. We give a short list. You should consult the documentation for more information on these commands.

sum, prod computes the sum and products of the rows
 cumsum, cumprod does the same cumulatively
 computes the extremal values of each row
 extrema returns a vector with the extremal information
 diag(A, i) returns the i-th diagonal
 setdiag(A, i, v) sets the i-th diagonal
 id(n) the identity matrix
 det(A) the determinant
 charpoly(A) the characteristic polynomial
 eigenvalues(A) the eigenvalues

```
>v*v, sum(v*v), cumsum(v*v)
```

```
[1, 4, 9]
14
[1, 5, 14]
```

The : operator generates an equally spaces row vector, optionally with a step size.

```
>1:4, 1:2:10
```

```
[1, 2, 3, 4]
[1, 3, 5, 7, 9]
```

To concatenate matrices and vectors there are the operators "|" and "_" .

```
>[1,2,3] | [4,5], [1,2,3]_1
```

```
[1, 2, 3, 4, 5]
1           2           3
1           1           1
```

The elements of a matrix are referred with "A[i,j]".

```
>A:=[1,2,3;4,5,6;7,8,9]; A[2,3]
```

For row or column vectors, v[i] is the i-th element of the vector. For matrices, this returns the complete i-th row of the matrix.

```
>v:=[2,4,6,8]; v[3], A[3]
```

```
6
[7, 8, 9]
```

The indices can also be row vectors of indices. : denotes all indices.

```
>v[1:2], A[:,2]
```

```
[2, 4]
2
5
8
```

A short form for : is omitting the index completely.

```
>A[,2:3]
```

```
2          3
5          6
8          9
```

For purposes of vectorization, the elements of a matrix can be accessed as if they were vectors.

```
>A{4}
```

```
4
```

A matrix can also be flattened, using the `redim()` function. This is implemented in the function `flatten()`.

```
>redim(A,1,prod(size(A))), flatten(A)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

To use matrices for tables, let us reset to the default format, and compute a table of sine and cosine values. Note that angles are in radians by default.

```
>deffformat; w=0°:45°:360°; w=w'; deg(w)
```

```
0
45
90
135
180
225
270
315
360
```

Now we append columns to a matrix.

```
>M = deg(w)|w|cos(w)|sin(w)
```

0	0	1	0
45	0.785398	0.707107	0.707107
90	1.5708	0	1
135	2.35619	-0.707107	0.707107
180	3.14159	-1	0
225	3.92699	-0.707107	-0.707107
270	4.71239	0	-1
315	5.49779	0.707107	-0.707107
360	6.28319	1	0

Using the matrix language, we can generate several tables of several functions at once.

In the following example, we compute $t[j]^i$ for i from 1 to n . We get a matrix, where each row is a table of t^i for one i . I.e., the matrix has the elements

$$a_{i,j} = t_j^i, \quad 1 \leq j \leq 101, \quad 1 \leq i \leq n$$

A function which does not work for vector input should be "vectorized". This can be achieved by the "map" keyword in the function definition. Then the function will be evaluated for each element of a vector parameter.

The numerical integration integrate() works only for scalar interval bounds. So we need to vectorize it.

```
>function map f(x) := integrate("x^x", 1, x)
```

The "map" keyword vectorizes the function. The function will now work for vectors of numbers.

```
>f([1:5])
```

[0, 2.05045, 13.7251, 113.336, 1241.03]

Sub-Matrices and Matrix-Elements

To access a matrix element, use the bracket notation.

```
>A=[1,2,3;4,5,6;7,8,9], A[2,2]
```

1	2	3
4	5	6
7	8	9
5		

We can access a complete line of a matrix.

```
>A[2]
```

[4, 5, 6]

In case of row or column vectors, this returns an element of the vector.

```
>v=1:3; v[2]
```

2

To make sure, you get the first row for a 1xn and a mxn matrix, specify all columns using an empty second index.

```
>A[2, ]
```

[4, 5, 6]

If the index is a vector of indices, Euler will return the corresponding rows of the matrix. Here we want the first and second row of A.

```
>A[ [1, 2] ]
```

1	2	3
4	5	6

We can even reorder A using vectors of indices. To be precise, we do not change A here, but compute a reordered version of A.

```
>A[ [3, 2, 1] ]
```

7	8	9
4	5	6
1	2	3

The index trick works with columns too.

This example selects all rows of A and the second and third column.

```
>A[ 1:3, 2:3 ]
```

2	3
5	6
8	9

For abbreviation ":" denotes all row or column indices.

```
>A[ :, 3 ]
```

3
6
9

Alternatively, leave the first index empty.

```
>A[ , 2:3 ]
```

2	3
5	6
8	9

We can also get the last line of A.

```
>A[-1]
```

[7, 8, 9]

Now let us change elements of A by assigning a submatrix of A to some value. This does in fact change the stored matrix A.

```
>A[1,1]=4
```

4	2	3
4	5	6
7	8	9

We can also assign a value to a row of A.

```
>A[1]=[-1,-1,-1]
```

-1	-1	-1
4	5	6
7	8	9

We can even assign to a sub-matrix if it has the proper size.

```
>A[1:2,1:2]=[5,6;7,8]
```

5	6	-1
7	8	6
7	8	9

Moreover, some shortcuts are allowed.

```
>A[1:2,1:2]=0
```

0	0	-1
0	0	6
7	8	9

A warning: Indices out of bounds return empty matrices, or an error message, depending on a system setting. The default is an error message. Remember, however, that negative indices may be used to access the elements of a matrix counting from the end.

```
>A[4]
```

Row index 4 out of bounds!

Error in:

A[4] ...

^

Sorting and Shuffling

The function `sort()` sorts a row vector.

```
>sort([5,6,4,8,1,9])
```

[1, 4, 5, 6, 8, 9]

It is often necessary to know the indices of the sorted vector in the original vector. This can be used to reorder another vector in the same way.

Let us shuffle a vector.

```
>v=shuffle(1:10)
```

[4, 5, 10, 6, 8, 9, 1, 7, 2, 3]

The indices contain the proper order of v.

```
>{vs,ind}=sort(v); v[ind]
```

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

This works for string vectors too.

```
>s=["a","d","e","a","aa","e"]
```

a
d
e
a
aa
e

```
>{ss,ind}=sort(s); ss
```

```
a
a
aa
d
e
e
```

As you see, the position of double entries is somewhat random.

```
>ind
```

```
[4, 1, 5, 2, 6, 3]
```

The function `unique` returns a sorted list of unique elements of a vector.

```
>intrandom(1,10,10), unique(%)
```

```
[4, 4, 9, 2, 6, 5, 10, 6, 5, 1]
[1, 2, 4, 5, 6, 9, 10]
```

This works for string vectors too.

```
>unique(s)
```

```
a
aa
d
e
```

Linear Algebra

EMT has lots of functions to solve linear systems, sparse systems, or regression problems. For linear systems $Ax=b$, you can use the Gauss algorithm, the inverse matrix or a linear fit. The operator $A\backslash b$ uses a version of the Gauss algorithm.

```
>A=[1,2;3,4]; b=[5;6]; A\b
```

```
-4
4.5
```

For another example, we generate a 200×200 matrix and the sum of its rows. Then we solve $Ax=b$ using the inverse matrix. We measure the error as the maximal deviation of all elements from 1, which of course is the correct solution.

```
>A=normal(200,200); b=sum(A); longest totalmax(abs(inv(A).b-1))
```

8.790745908981989e-13

If the system does not have a solution, a linear fit minimizes the norm of the error $Ax-b$.

```
>A=[1,2,3;4,5,6;7,8,9]
```

1	2	3
4	5	6
7	8	9

The determinant of this matrix is 0.

```
>det(A)
```

0

Symbolic Matrices

Maxima has symbolic matrices. Of course, Maxima can be used for such simple linear algebra problems. We can define the matrix for Euler and Maxima with &:=, and then use it in symbolic expressions. The usual [...] form to define matrices can be used in Euler to define symbolic matrices.

```
>A &= [a,1,1;1,a,1;1,1,a]; $A
>$&det(A), $&factor(%)
>$&invert(A) with a=0
>A &= [1,a;b,2]; $A
```

Like all symbolic variables, these matrices can be used in other symbolic expressions.

```
>$&det(A-x*ident(2)), $&solve(% ,x)
```

The eigenvalues can also be computed automatically. The result is a vector with two vectors of eigenvalues and multiplicities.

```
>${&eigenvalues([a,1;1,a])}
```

To extract a specific eigenvector needs careful indexing.

```
>${&eigenvectors([a,1;1,a]), &%[2][1][1]
```

```
[1, - 1]
```

Symbolic matrices can be evaluated in Euler numerically just like other symbolic expressions.

```
>A(a=4,b=5)
```

1	4
5	2

In symbolic expressions, use with.

```
>${&A with [a=4,b=5]}
```

Access to rows of symbolic matrices work just like with numerical matrices.

```
>${&A[1]}
```

A symbolic expression can contain an assignment. And that changes the matrix A.

```
>&A[1,1]:=t+1; ${&A}
```

There are symbolic functions in Maxima to create vectors and matrices. For this, refer to the documentation of Maxima or to the tutorial about Maxima in EMT.

```
>v &= makelist(1/(i+j), i, 1, 3); $v
```

```
>B &:= [1,2;3,4]; $B, $&invert(B)
```

The result can be evaluated numerically in Euler. For more information about Maxima, see the introduction to Maxima.

```
>$&invert(B)()
```

-2	1
1.5	-0.5

Euler has also a powerful function `xinv()`, which makes a bigger effort and gets more exact results.

Note, that with `&:=` the matrix B has been defined as symbolic in symbolic expressions and as numerical in numerical expressions. So we can use it here.

```
>longest B.xinv(B)
```

1	0
0	1

E.g. the eigenvalues of A can be computed numerically.

```
>A=[1,2,3;4,5,6;7,8,9]; real(eigenvalues(A))
```

[16.1168, -1.11684, 0]

Or symbolically. See the tutorial about Maxima for details on this.

```
>$&eigenvalues(@A)
```

Numerical Values in symbolic Expressions

A symbolic expression is just a string containing an expression. If we want to define a value both for symbolic expressions and for numerical expressions, we must use "`&:=`".

```
>A &:= [1,pi;4,5]
```

1	3.14159
4	5

There is still a difference between the numerical and the symbolic form. When transferring the matrix to the symbolic form, fractional approximations for reals will be used.

```
> $&A
```

To avoid this, there is the function "mxmset(variable)".

```
>mxmset (A); $&A
```

Maxima can also compute with floating point numbers, and even with big floating numbers with 32 digits. The evaluation is much slower, however.

```
>$&bfloat(sqrt(2)), $&float(sqrt(2))
```

The precision of the big floating point numbers can be changed.

```
>&fpprec:=100; &bfloat(pi)
```

```
3.14159265358979323846264338327950288419716939937510582097494\
4592307816406286208998628034825342117068b0
```

A numerical variable can be used in any symbolic expressions using "@var".

Note that this is only necessary, if the variable has been defined with ":=" or "=" as a numerical variable.

```
>B:=[1,pi;3,4]; $&det (@B)
```

Demo - Interest Rates

Below, we use Euler Math Toolbox (EMT) for the calculation of interest rates. We do that numerically and symbolically to show you how Euler can be used to solve real life problems. Assume you have a seed capital of 5000 (say in dollars).

```
>K=5000
```

```
5000
```

Now we assume an interest rate of 3% per year. Let us add one simple rate and compute the result.

```
>K*1.03
```

5150

Euler would understand the following syntax too.

```
>K+K*3%
```

5150

But it is easier to use the factor

```
>q=1+3%, K*q
```

1.03

5150

For 10 years, we can simply multiply the factors and get the final value with compound interest rates.

```
>K*q^10
```

6719.58189672

For our purposes, we can set the format to 2 digits after the decimal dot.

```
>format(12,2); K*q^10
```

6719.58

Let us print that rounded to 2 digits in a complete sentence.

```
>"Starting from " + K + "$ you get " + round(K*q^10,2) + "$."
```

Starting from 5000\$ you get 6719.58\$.

What if we want to know the intermediate results from year 1 to year 9? For this, Euler's matrix language is a big help. You do not have to write a loop, but simply enter

```
>K*q^(0:10)
```

Real 1 x 11 matrix

5000.00	5150.00	5304.50	5463.64	...
---------	---------	---------	---------	-----

How does this miracle work? First the expression 0:10 returns a vector of integers.

```
>short 0:10
```

[0,	1,	2,	3,	4,	5,	6,	7,	8,	9,	10]
-----	----	----	----	----	----	----	----	----	----	-----

Then all operators and functions in Euler can be applied to vectors element for element. So

```
>short q^(0:10)
```

[1,	1.03,	1.0609,	1.0927,	1.1255,	1.1593,	1.1941,	1.2299,
1.2668,	1.3048,	1.3439]					

is a vector of factors q^0 to q^{10} . This is multiplied by K, and we get the vector of values.

```
>VK=K*q^(0:10);
```

Of course, the realistic way to compute these interest rates would be to round to the nearest cent after each year. Let us add a function for this.

```
>function oneyear (K) := round(K*q, 2)
```

Let us compare the two results, with and without rounding.

```
>longest oneyear(1234.57), longest 1234.57*q
```

1271.61
1271.6071

Now there is no simple formula for the n-th year, and we must loop over the years. Euler provides many solutions for this.

The easiest way is the function iterate, which iterates a given function a number of times.

```
>VKr=iterate("oneyear",5000,10)
```

Real 1 x 11 matrix

5000.00	5150.00	5304.50	5463.64	...
---------	---------	---------	---------	-----

We can print that in a friendly way, using our format with fixed decimal places.

```
>VKr'
```

5000.00
5150.00
5304.50
5463.64
5627.55
5796.38
5970.27
6149.38
6333.86
6523.88
6719.60

To get a specific element of the vector, we use indices in square brackets.

```
>VKr[2], VKr[1:3]
```

5150.00
5000.00
5150.00
5304.50

Surprisingly, we can also use a vector of indices. Remember that 1:3 produced the vector [1,2,3].

Let us compare the last element of the rounded values with the full values.

```
>VKr[-1], VK[-1]
```

6719.60
6719.58

The difference is very small.

Solving Equations

Now we take a more advanced function, which adds a certain rate of money each year.

```
>function onepay (K) := K*q+R
```

We do not have to specify q or R for the definition of the function. Only if we run the command, we have to define these values. We select R=200.

```
>R=200; iterate("onepay",5000,10)
```

Real 1 x 11 matrix

5000.00	5350.00	5710.50	6081.82	...
---------	---------	---------	---------	-----

What if we remove the same amount each year?

```
>R=-200; iterate("onepay",5000,10)
```

Real 1 x 11 matrix

5000.00	4950.00	4898.50	4845.45	...
---------	---------	---------	---------	-----

We see that the money decreases. Obviously, if we get only 150 of interest in the first year, but remove 200, we lose money each year.

How can we determine the number of years the money will last? We would have to write a loop for this. The easiest way is to iterate long enough.

```
>VKR=iterate("onepay",5000,50)
```

Real 1 x 51 matrix

5000.00	4950.00	4898.50	4845.45	...
---------	---------	---------	---------	-----

Using the matrix language, we can determine the first negative value in the following way.

```
>min(nonzeros(VKR<0))
```

48.00

The reason for this is that nonzeros(VKR<0) returns a vector of indices i, where VKR[i]<0, and min computes the minimal index.

Since vectors always start with index 1, the answer is 47 years.

The function iterate() has one more trick. It can take an end condition as an argument. Then it will return the value and the number of iterations.

```
>{x,n}=iterate("onipay",5000,till="x<0"); x, n,
```

-19.83

47.00

Let us try to answer a more ambiguous question. Assume we know that the value is 0 after 50 years. What would be the interest rate?

This is a question, which can only be answered numerically. Below, we will derive the necessary formulas. Then you will see that there is no easy formula for the interest rate. But for now, we aim for a numerical solution.

The first step is to define a function which does the iteration n times. We add all parameters to this function.

```
>function f(K,R,P,n) := iterate("x*(1+P/100)+R",K,n;P,R)[-1]
```

The iteration is just as above

$$x_{n+1} = x_n \cdot \left(1 + \frac{P}{100}\right) + R$$

But we do longer use the global value of R in our expression. Functions like iterate() have a special trick in Euler. You can pass the values of variables in the expression as semicolon parameters. In this case P and R.

Moreover, we are only interested in the last value. So we take the index [-1].

Let us try a test.

```
>f(5000,-200,3,47)
```

-19.83

Now we can solve our problem.

```
>solve("f(5000,-200,x,50)",3)
```

3.15

The solve routine solves expression=0 for the variable x. The answer is 3.15% per year. We take the start value of 3% for the algorithm. The solve() function always needs a start value. We can use the same function to solve the following question: How much can we remove per year so that the seed capital is exhausted after 20 years assuming an interest rate of 3% per year.

```
>solve("f(5000,x,3,20)",-200)
```

-336.08

Note that you cannot solve for the number of years, since our function assumes n to be an integer value.

Solusi Simbolis Masalah Suku Bunga

Kita dapat menggunakan bagian simbolis dari Euler untuk mempelajari masalahnya. Pertama kita mendefinisikan fungsi onepay() kita secara simbolis.

```
>function op(K) &= K*q+R; $&op(K)
```

$$qK + R$$

Sekarang kita dapat mengulanginya.

```
>$&op(op(op(op(K)))) , $&expand
```

$$R + q(R + q(R + q(qK + R)))$$

expand

Kita melihat sebuah pola. Setelah n periode yang kita miliki

$$Kn = q^n K + R(1 + q + \dots + q^{n-1}) = q^n K + \frac{q^n - 1}{q - 1} R$$

Rumusnya adalah rumus jumlah geometri yang diketahui Maxima. Untuk mempelajari masalahnya. Pertama kita mendefinisikan fungsi onepay() kita secara simbolis.

```
>&sum(q^k, k, 0, n-1); $& % = ev(% , simpsum)
```

$$\sum_{k=0}^{-1+n} q^k = \frac{-1 + q^n}{-1 + q}$$

Ini agak rumit. Jumlahnya dievaluasi dengan tanda "simpsum" untuk menguranginya menjadi hasil bagi. Mari kita membuat fungsi untuk ini.

```
>function fs(K, R, P, n) &= (1+P/100)^n*K + ((1+P/100)^n-1)/(P/100)*R; $&fs(K,
```

$$K \left(1 + \frac{P}{100}\right)^n + \frac{100 \left(-1 + \left(1 + \frac{P}{100}\right)^n\right) R}{P}$$

Fungsinya sama dengan fungsi f kita sebelumnya. Tapi ini lebih efektif.

```
>longest f(5000,-200,3,47), longest fs(5000,-200,3,47)
```

```
Function f not found.  
Try list ... to find functions!  
Error in:  
longest f(5000,-200,3,47), longest fs(5000,-200,3,47) ...  
^
```

Sekarang kita dapat menggunakan untuk menanyakan waktu n. Kapan modal kita habis? Perkiraan awal kami adalah 30 tahun.

```
>solve("fs(5000,-330,3,x)",30)
```

20.51

Jawaban ini mengatakan akan menjadi negatif setelah 21 tahun.

Kita juga dapat menggunakan sisi simbolis Euler untuk menghitung rumus pembayaran. Asumsikan kita mendapatkan pinjaman sebesar K, dan membayar n pembayaran sebesar R (dimulai setelah tahun pertama) meninggalkan sisa hutang sebesar Kn (pada saat pembayaran terakhir). Rumusnya jelas

```
>equ &= fs(K,R,P,n)=Kn; $&equ
```

$$fs(K, R, P, n) = Kn$$

Biasanya rumus ini diberikan dalam bentuk:

$$i = \frac{P}{100}$$

```
>equ &= (equ with P=100*i); $&equ
```

equ

Kita dapat menyelesaikan nilai R secara simbolis.

```
>$&solve(equ, R)
```

□

Seperti yang Anda lihat dari rumusnya, fungsi ini mengembalikan kesalahan floating point untuk $i=0$. Euler tetap merencanakannya.

entu saja kita memiliki limit berikut

```
>$&limit(R(5000, 0, x, 10), x, 0)
```

$$\lim_{x \rightarrow 0} R(5000, 0, x, 10)$$

Yang jelas tanpa bunga kita harus membayar kembali 10 tarif 500.

Persamaan tersebut juga dapat diselesaikan untuk n. Akan terlihat lebih bagus jika kita menerapkan beberapa penyederhanaan padanya.

```
>fn &= solve(equ, n) | ratsimp; $&fn
```

□

Latihan Soal R2

88
Soal No. 49

BAB 2. EMT UNTUK PERHITUNGAN ALJABAR

$$\left(\frac{24a^{10}b^{-8}c^7}{12a^6b^{-3}c^5} \right)^{-5}$$

$$>\$ \& ((24*a^{10}*b^{-8}*c^7) / (12*a^6*b^{-3}*c^5))^{-5}$$

$$\frac{b^{25}}{32 a^{20} c^{10}}$$

Soal No. 50

$$>\$ \& ((125*p^{12}*q^{-14}*r^{22}) / (25*p^8*q^6*r^{-15}))^{-4}$$

$$\frac{q^{80}}{625 p^{16} r^{148}}$$

$$>\$ \& (2^6 2^{-3} / (2^{10}) / (2^{-8}))$$

2

Soal No. 91

$$>\$ \& (4 * (8-6)^2 - 4 * 3 + 2 * 8) / (3^1 + 9^0)$$

5

Soal No. 92

$$>\$ \& ((4 * (8-6)^2 + 4) * (3 - 2 * 8)) / (2^2 * (2^3 + 5))$$

-5

Soal No. 104

> \$& ((m^(x-b)*n^(x+b)^x)*(m^b*n^(-b))^x)

$$m^{x-b} \left(\frac{m^b}{n^b}\right)^x n^x (x+b)$$

Latihan Soal R3

Soal No. 9

> \$& ((3*x^2 - 2*x - x^3 + 2) - (5*x^2 - 8*x - x^3 + 4))

$$-2x^2 + 6x - 2$$

Soal No. 13

> \$& ((3*a^2) * ((-7) * a^4))

$$-21a^6$$

Soal No. 14

> \$& ((8*y^5) * (9*y))

$$72y^6$$

Soal No. 15

> \$& ((6*x*y^3) * (9*x^4*y^2))

$$54x^5y^5$$

Soal. No 36

```
> $% ( (4*x^2 - 5*y)^2)
```

$$(54x^5y^5)((4x^2 - 5y)^2)$$

Latihan Soal R4

Soal No. 23

```
> $& factor(t^2 + 8*t + 15)
```

$$(t + 3) (t + 5)$$

Soal No. 24

```
> $& factor(y^2 + 12*y + 27)
```

$$(y + 3) (y + 9)$$

Soal No. 121

```
> $& factor(y^4 - 84 + 5*y^2)
```

$$(y^2 - 7) (y^2 + 12)$$

Soal No. 77

```
> $& factor((18*a^2*b) - (15*a*b^2))
```

$$-3ab(5b - 6a)$$

Soal No. 78

```
>${&} factor(4*x^2*y+12*x*y^2)
```

$$4xy(3y+x)$$

Latihan Soal R5

Soal No. 31

```
>${&} solve(7*(3*x+6)= 11-(x+2))
```

$$\left[x = -\frac{3}{2} \right]$$

Soal No. 32

```
>${&} solve(9*(2*x+8)= 20-(x-5))
```

$$\left[x = -\frac{47}{19} \right]$$

Soal No. 35

```
>${&} solve(x^2+3*x-28=0)
```

$$[x = 4, x = -7]$$

Soal No. 36

```
>${&} solve(y^2-4*y-45=0)
```

$$[y = 9, y = -5]$$

Soal No. 37

```
>${\&} solve(x^2+5*x=0)
```

$$[x = -5, x = 0]$$

Latihan Soal R6

Soal No. 9

```
>${\&} ratsimp((x^2-4)/(x^2-4*x+4))
```

$$\frac{x + 2}{x - 2}$$

Soal No. 11

```
>${\&} ratsimp((x^3-6*x^2+9*x)/(x^3-3*x^2))
```

$$\frac{x - 3}{x}$$

Soal No. 14

```
>${\&} ratsimp((2*x^2-20*x+50)/(10*x^2-30*x-100))
```

$$\frac{x - 5}{5x + 10}$$

Soal No. 15

```
>${\&} ratsimp((6-x)/(x^2-36))
```

$$-\frac{1}{x + 6}$$

Soal No. 20

```
> $& ratsimp( ((x^2-2*x-35)/(2*x^3-3*x^2)) * ((4*x^3-9*x)/(7*x-49)) )
```

$$\frac{2x^2 + 13x + 15}{7x}$$

Latihan soal 3.1

Soal No. 11

```
> $ powerdisp: true
```

true

```
> $& ( (-5+3*I) + (7+8*I) )
```

$$2 + 11i$$

Soal No. 31

```
> $& (sqrt(-4)*(sqrt(-36)))
```

$$-12$$

Soal No. 53

```
> $& solve(2*x^2+1=5*x)
```

$$\left[x = \frac{5 - \sqrt{17}}{4}, x = \frac{\sqrt{17} + 5}{4} \right]$$

Soal No. 84

```
> $& solve(y^4-15*y^2-16=0)
```

$$[y = -i, y = i, y = -4, y = 4]$$

```
>$& solve(x^4-3*x^2+2=0)
```

$$\left[x = -\sqrt{2}, x = \sqrt{2}, x = -1, x = 1 \right]$$

Latihan Soal 3.4

Soal No. 01

```
>$& solve((1/4)+(1/5)=(1/t))
```

$$\left[t = \frac{20}{9} \right]$$

Soal No. 9

```
>$& solve(x+(6/x)=5)
```

$$[x = 3, x = 2]$$

Soal No. 15

```
>$& solve((2/(x+5))+(1/(x-5))=(16/(x^2-25)))
```

$$[x = 7]$$

Soal No. 29

```
>$& solve(sqrt(3*x-4)=1)
```

$$\left[x = \frac{5}{3} \right]$$

Soal No. 33

```
>$& solve(sqrt(2*x-5)=2)
```

$$\left[x = \frac{9}{2} \right]$$

Latihan Soal 3.5

Soal NO. 23

```
>$& solve(abs(x+3)-2=8, [x])
```

$$|x + 3| = 10$$

Soal No. 24

```
>$& fourier_elim(abs(x-4)+3=9, [x])
```

$$\text{fourier_elim } (|x - 4| + 3 = 9, [x])$$

Soal No. 25

```
>$& fourier_elim(abs(3*x+1)-4=(-1), [x])
```

$$\text{fourier_elim } (|3x + 1| - 4 = -1, [x])$$

Soal No. 32

```
>$& solve(5-abs(4*x+3)=2)
```

$$|4x + 3| = 3$$

Soal No. 30

```
> $& fourier_elim(9-abs(x-2)=7, [x])
```

$$\text{fourier_elim} (9 - |x - 2| = 7, [x])$$

Latihan Soal 4.3

Soal No. 70

```
>
> $ powerdisp: true
```

true

```
> $& ( ratsimp((x^4-y^4)/(x-y)) )
```

$$x^3 + x^2 y + x y^2 + y^3$$

Soal No. 19

```
> $& ratsimp((x^4-1)/(x-1))
```

$$x^3 + x^2 + x + 1$$

Soal No. 17

```
> $ powerdisp:true
```

true

```
> $& ratsimp((x^5+x^3-x)/(x-3))
```

$$\frac{-x + x^3 + x^5}{-3 + x}$$

Soal No. 22

```
>${\&} solve(ratsimp((3*x^2-2*x^2+2)/(x-(1/4)))
```

$$\left[x = -\sqrt{2}i, x = \sqrt{2}i \right]$$

Soal No. 11

```
>${\&} ratsimp((2*x^4+7*x^3+x-12)/(x+3))
```

$$\frac{-12 + x + 7x^3 + 2x^4}{3 + x}$$

Latihan Soal R7

Soal No. 76

```
>${\&} factor(m^(6*n)-m^(3*n))
```

$$m^{3n} (-1 + m^n) (1 + m^n + m^{2n})$$

Soal No. 70

```
>${\&} expand(((x^n)+10)*(x^n)-4))
```

$$-40 + 6x^n + x^{2n}$$

Soal No. 75

```
>${\&} factor(x^(2*t)-3*x^t-28)
```

$$(-7 + x^t) (4 + x^t)$$

Soal No. 74

```
> $& factor(y^(2*n) + 16*y^n + 64)
```

$$(8 + y^n)^2$$

Soal No. 73

```
> $& expand((a^n - b^n)^3)
```

$$a^{3n} - 3a^{2n}b^n + 3a^n b^{2n} - b^{3n}$$

BAB

Menggambar Grafik 2D dengan EMT

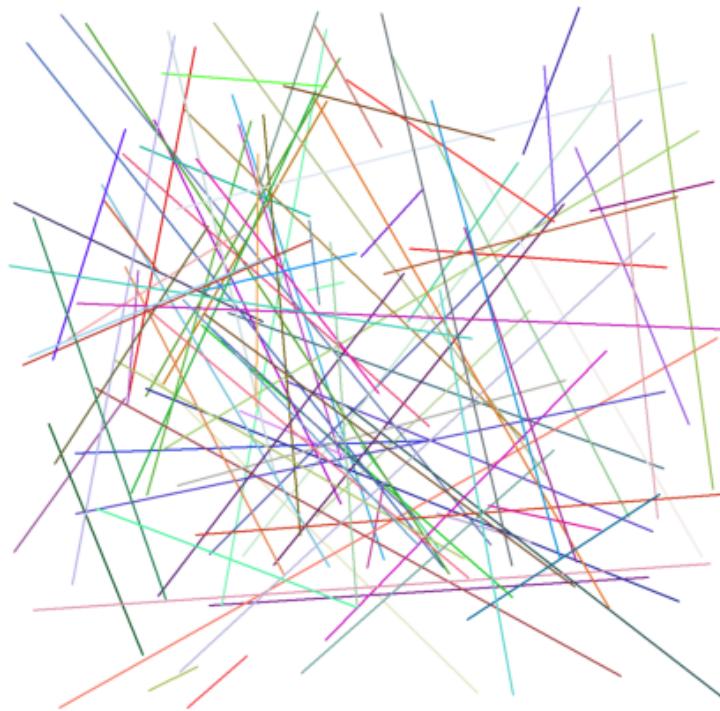
Notebook ini menjelaskan tentang cara menggambar berbagai kurva dan grafik 2D dengan software EMT. EMT menyediakan fungsi plot2d() untuk menggambar berbagai kurva dan grafik dua dimensi (2D).

Basic Plots (Plot Dasar)

Ada fungsi plot yang sangat mendasar. Terdapat koordinat layar, yang selalu berkisar dari 0 hingga 1024 di setiap sumbu, tidak peduli apakah layarnya persegi atau tidak. Terdapat koordinat plot, yang dapat diatur dengan setplot(). Pemetaan antara koordinat tergantung pada jendela plot saat ini. Sebagai contoh, default shrinkwindow() menyisakan ruang untuk label sumbu dan judul plot.

Dalam contoh, kita hanya menggambar beberapa garis acak dalam berbagai warna. Untuk detail mengenai fungsi-fungsi ini, pelajari fungsi inti EMT.

```
>clg; // clear screen
>window(0,0,1024,1024); // use all of the window
>setplot(0,1,0,1); // set plot coordinates
>hold on; // start overwrite mode
>n=100; X=random(n,2); Y=random(n,2); // get random points
>colors=rgb(random(n),random(n),random(n)); // get random colors
>loop 1 to n; color(colors[#]); plot(X[#],Y[#]); end; // plot
>hold off; // end overwrite mode
>insimg; // insert to notebook
```



```
>reset;
```

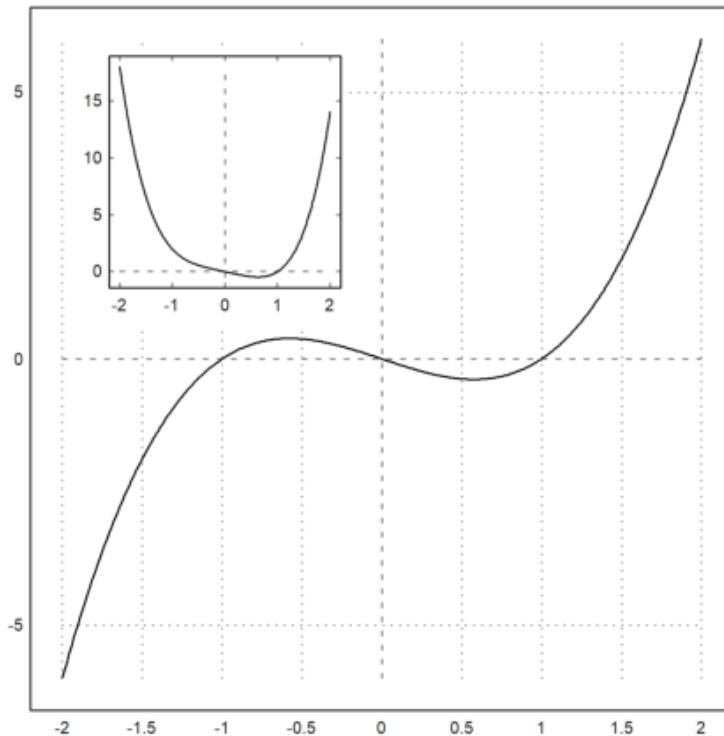
Anda harus menahan grafik, karena perintah `plot()` akan menghapus jendela plot. Untuk menghapus semua yang telah kita lakukan, kita menggunakan `reset()`.

Untuk menampilkan gambar hasil plot di layar notebook, perintah `plot2d()` dapat diakhiri dengan titik dua(:).

Cara lain untuk perintah `plot2d()` diakhiri dengan titik koma(;), kemudian gunakan perintah `insimg()` untuk menampilkan gambar hasil plot.

Sebagai contoh lain, kita menggambar plot sebagai inset dalam plot lain. Hal ini dilakukan dengan mendefinisikan jendela plot yang lebih kecil. Perhatikan bahwa jendela ini tidak menyediakan ruang untuk label sumbu di luar jendela plot. Kita harus menambahkan beberapa margin untuk hal ini sesuai kebutuhan. Perhatikan bahwa kita menyimpan dan mengembalikan jendela penuh, dan menahan plot saat ini sementara kita membuat inset.

```
>plot2d("x^3-x");
>xw=200; yw=100; ww=300; hw=300;
>ow>window();
>>window(xw,yw,xw+ww,yw+hw);
>hold on;
>barclear(xw-50,yw-10,ww+60,ww+60);
>plot2d("x^4-x",grid=6):
```



```
>hold off;
>window(ow);
```

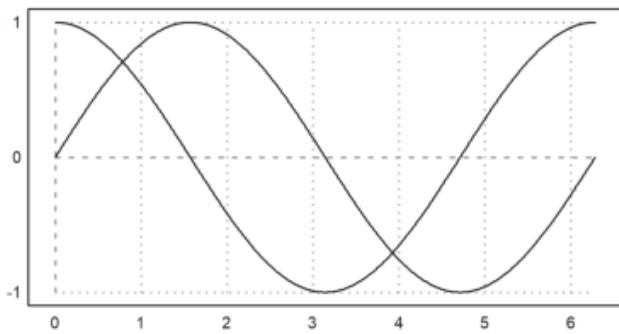
Plot dengan beberapa angka dicapai dengan cara yang sama. Ada fungsi utility figure() untuk ini.

Plot Aspect (Aspek plot)

Plot default menggunakan jendela plot persegi. Anda dapat mengubahnya dengan fungsi aspect(). Jangan lupa untuk mengatur ulang aspeknya nanti. Anda juga bisa mengubah default ini di menu "Set Aspect" ke rasio aspek tertentu atau ke ukuran jendela grafis saat ini.

Tetapi Anda juga dapat mengubahnya untuk satu plot. Untuk melakukan ini, ukuran area plot saat ini diubah, dan jendela diatur sedemikian rupa sehingga label memiliki ruang yang cukup.

```
>aspect(2); // rasio panjang dan lebar 2:1
>plot2d(["sin(x)", "cos(x")], 0, 2pi);
```



```
>aspect();
>reset;
```

Fungsi reset() memulihkan default plot, termasuk rasio aspek.

2D Plots in Euler

EMT Math Toolbox memiliki plot dalam bentuk 2D, baik untuk data maupun fungsi. EMT menggunakan fungsi plot2d. Fungsi ini dapat memplot fungsi dan data.

Hal ini memungkinkan untuk memplot di Maxima menggunakan Gnuplot atau di Python menggunakan Math Plot Lib.

Euler dapat memplot plot 2D dari:

- ekspresi
- fungsi, variabel, atau kurva yang diparameterkan,
- vektor nilai x-y,
- awan titik-titik di dalam pesawat,
- kurva implisit dengan level atau wilayah level,
- Fungsi yang kompleks

Gaya plot mencakup berbagai gaya untuk garis dan titik, plot batang dan plot berbayang.

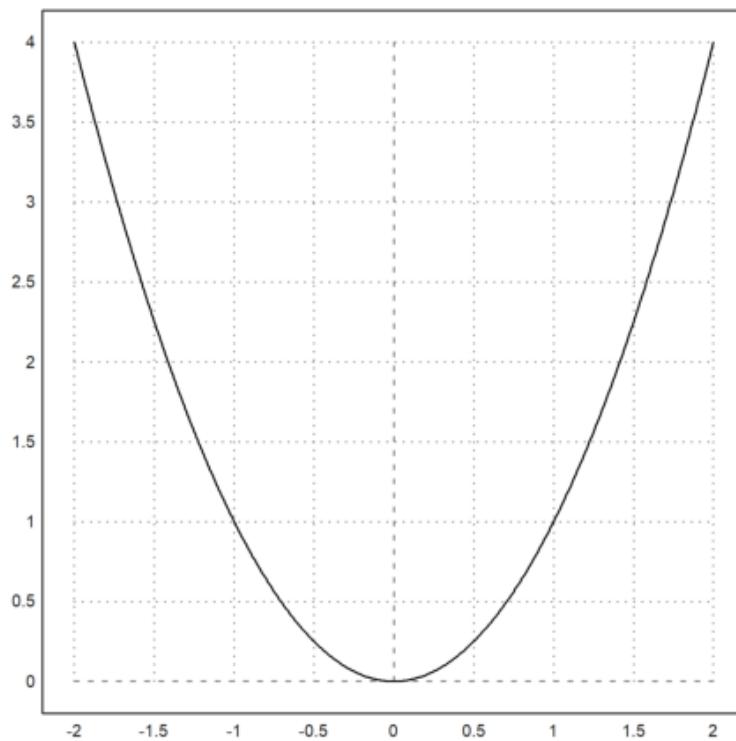
Plot Ekspresi atau Variabel

Sebuah ekspresi tunggal dalam "x" (misalnya "4*x^2) atau nama fungsi (misalnya "f") menghasilkan grafik fungsi.

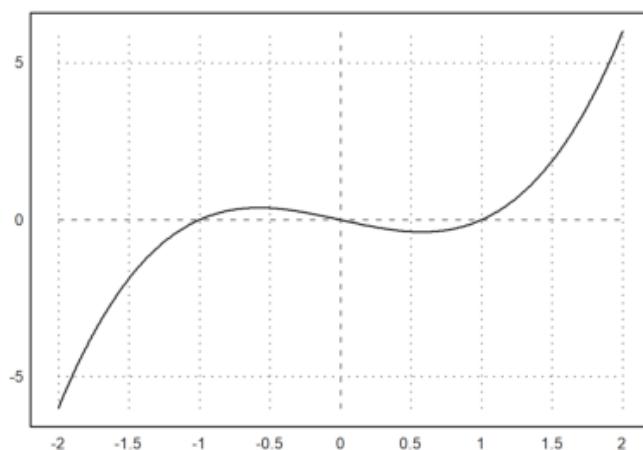
Berikut ini adalah contoh paling dasar, yang menggunakan rentang default dan menetapkan rentang y yang tepat agar sesuai dengan plot fungsi.

Catatan: Jika anda mengakhiri baris perintah dengan tanda titik dua "...", plot akan disisipkan ke dalam jendela teks. Jika tidak, tekan TAB untuk melihat plot jika jendela plot tertutup.

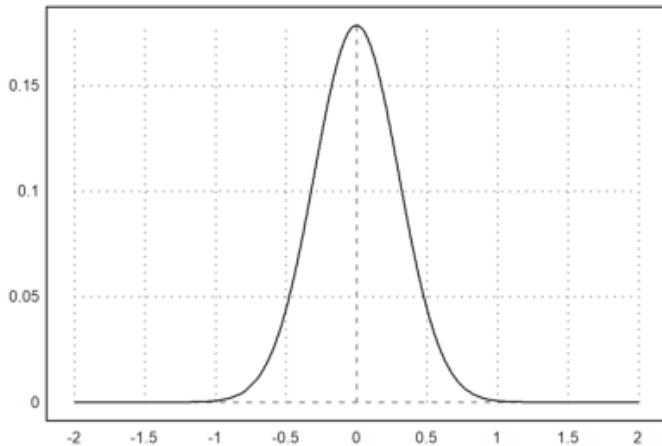
```
>plot2d("x^2"):
```



```
>aspect(1.5); plot2d("x^3-x"):
```



```
>a:=5.6; plot2d("exp(-a*x^2)/a"); insimg(30); // menampilkan gambar hasil p
```

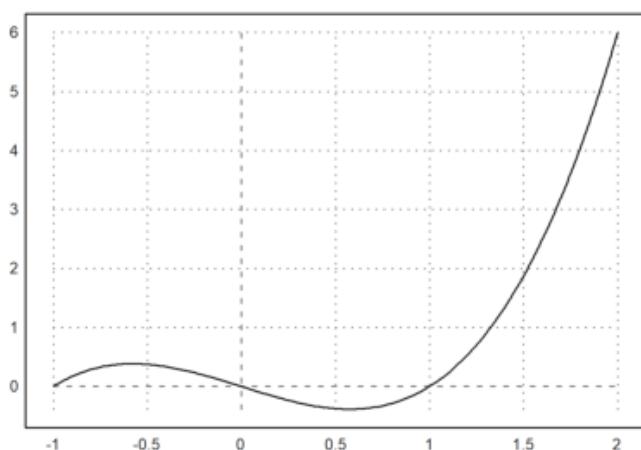


Dari beberapa contoh sebelumnya Anda dapat melihat bahwa aslinya gambar plot menggunakan sumbu X dengan rentang nilai dari -2 sampai dengan 2. Untuk mengubah rentang nilai X dan Y, Anda dapat menambahkan nilai-nilai batas X (dan Y) di belakang ekspresi yang digambar.

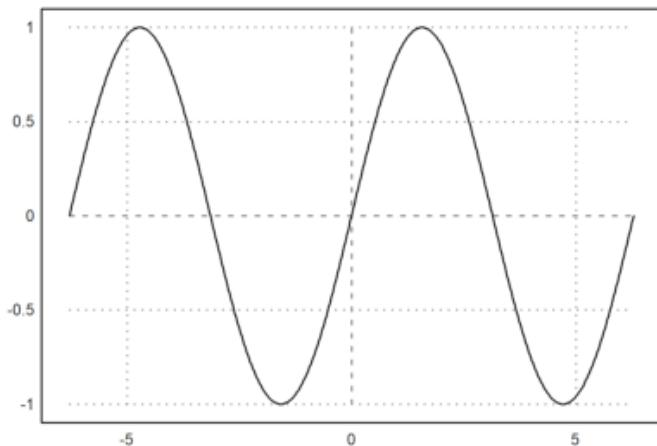
Rentang plot ditetapkan dengan parameter yang ditetapkan berikut ini:

- a,b: rentang-x (default -2,2)
- c,d: rentang-y (default: skala dengan nilai)
- r: sebagai alternatif adalah radius di sekitar pusat plot
- cx,cy: koordinat pusat plot (standar 0,0)

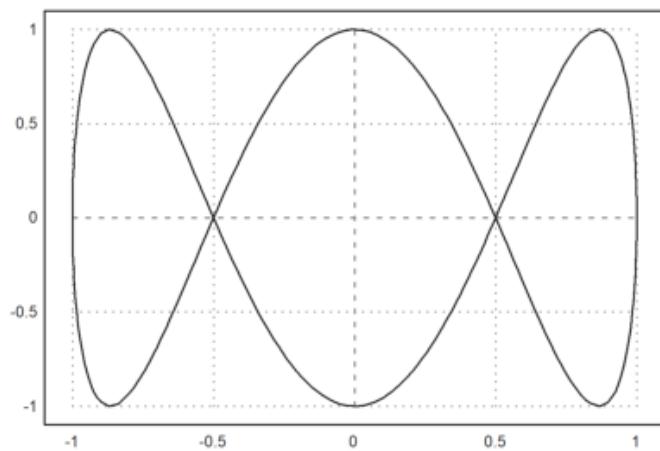
```
>plot2d("x^3-x", -1, 2):
```



```
>plot2d("sin(x)", -2*pi, 2*pi): // plot sin(x) pada interval [-2pi, 2pi]
```



```
>plot2d("cos(x)","sin(3*x)",xmin=0,xmax=2pi):
```



Alternatif untuk tanda titik dua adalah perintah insimg(lines), yang menyisipkan plot yang menempati sejumlah baris teks tertentu.

Dalam "options", plot dapat diatur untuk muncul

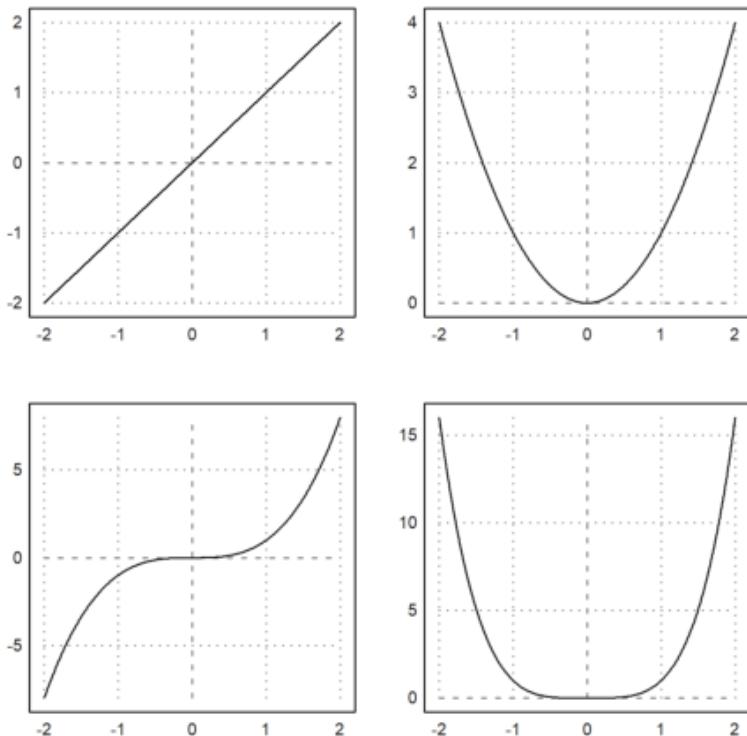
- dalam jendela terpisah yang dapat diubah ukurannya,
- di jendela notebook.

Lebih banyak gaya dapat dicapai dengan perintah plot tertentu.

Dalam hal apapun, tekan tombol tabulator untuk melihat plot, jika disembunyikan.

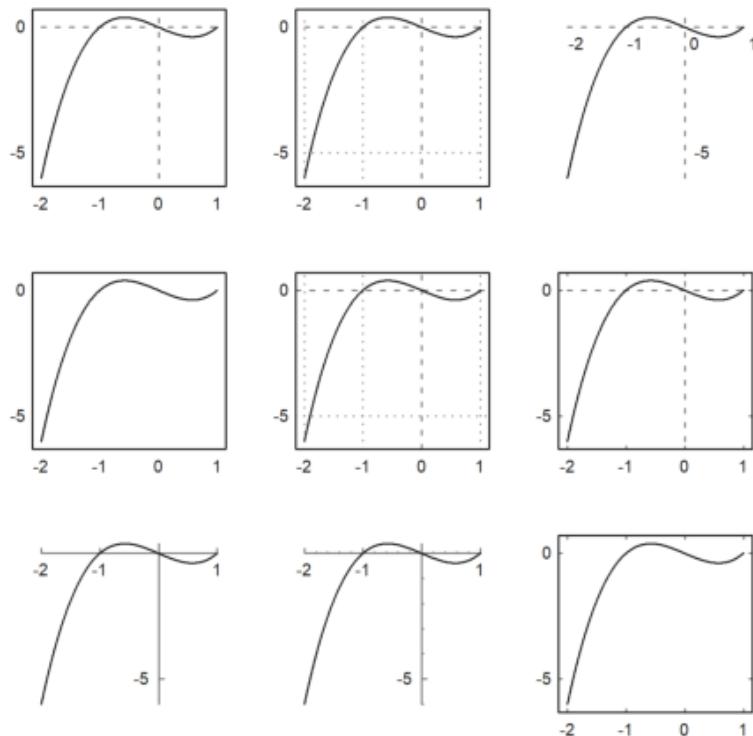
Untuk membagi jendela menjadi beberapa plot, gunakan perintah figure(). Pada contoh, kita memplot x^1 hingga x^4 menjadi 4 bagian jendela, figure(0) mengatur ulang jendela default.

```
>reset;
>figure(2,2); ...
>for n=1 to 4; figure(n); plot2d("x^"+n); end; ...
>figure(0):
```



In `plot2d()`, there are alternative styles available with `grid=x`. For an overview, we show the various grid styles in one figure (see below for the `figure()` command). The style `grid=0` is not included. It shows no grid and no frame.

```
>figure(3,3); ...
>for k=1:9; figure(k); plot2d("x^3-x",-2,1,grid=k); end; ...
>figure(0):
```

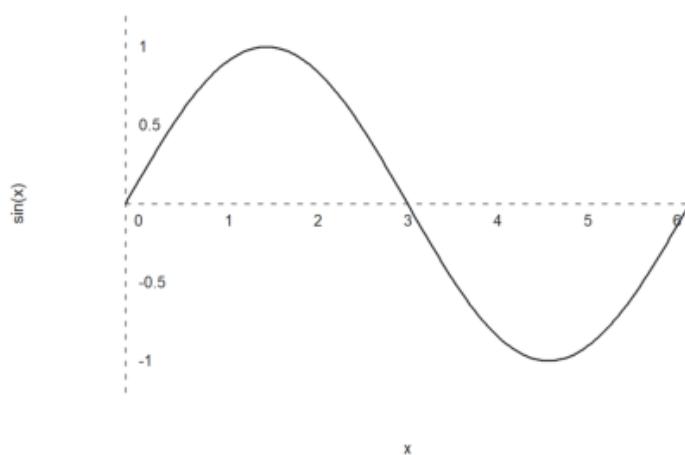


If the arguments to `plot2d()` are an expression followed by four numbers, these numbers are the x- and y-ranges for the plot.

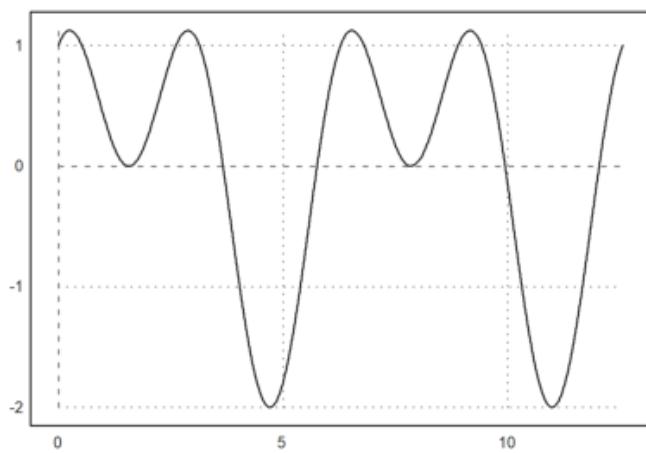
Alternatively, a, b, c, d can be specified as assigned parameters as $a=...$ etc.

In the following example, we change the grid style, add labels, and use vertical labels for the y-axis.

```
>aspect(1.5); plot2d("sin(x)", 0, 2pi, -1.2, 1.2, grid=3, xl="x", yl="sin(x)":
```



```
>plot2d("sin(x)+cos(2*x)", 0, 4pi):
```

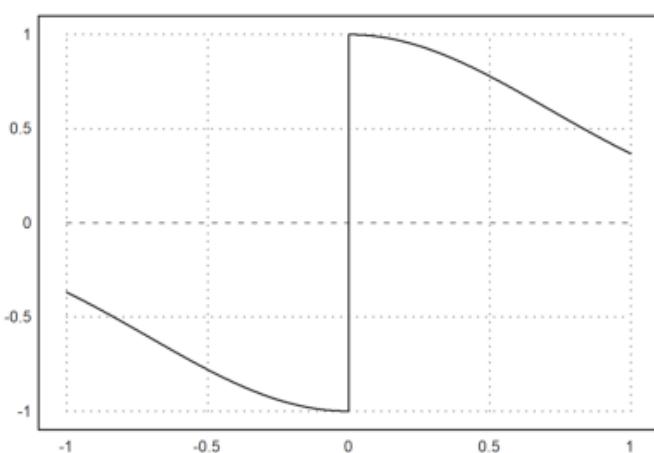


The images generated by inserting the plot into the text window are stored in the same directory as the notebook, by default in a subdirectory named "images". They are also used by the HTML export.

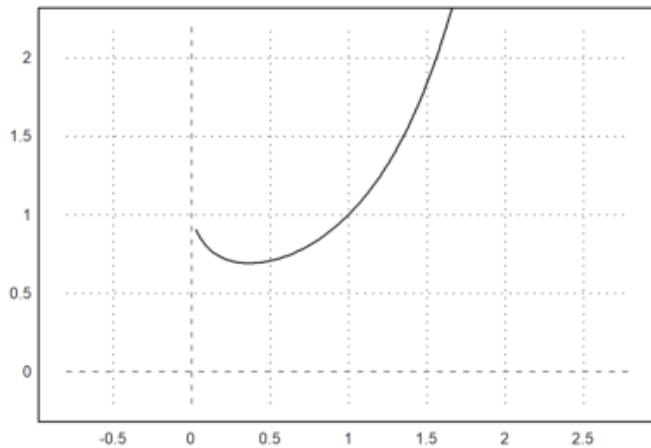
You can simply mark any image and copy it to the clipboard with Ctrl-C. Of course, you can also export the current graphics with the functions in the File menu.

The function or the expression in plot2d is evaluated adaptively. For more speed, switch off adaptive plots with <adaptive and specify the number of subintervals with n=... This should be necessary in rare cases only.

```
>plot2d("sign(x)*exp(-x^2)", -1, 1, <adaptive, n=10000):
```

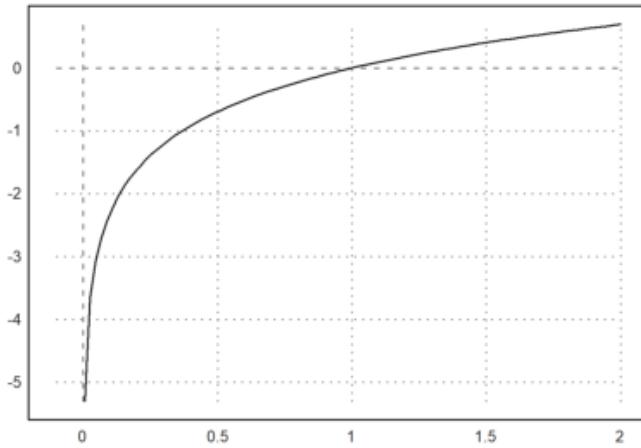


```
>plot2d("x^x", r=1..2, cx=1, cy=1):
```



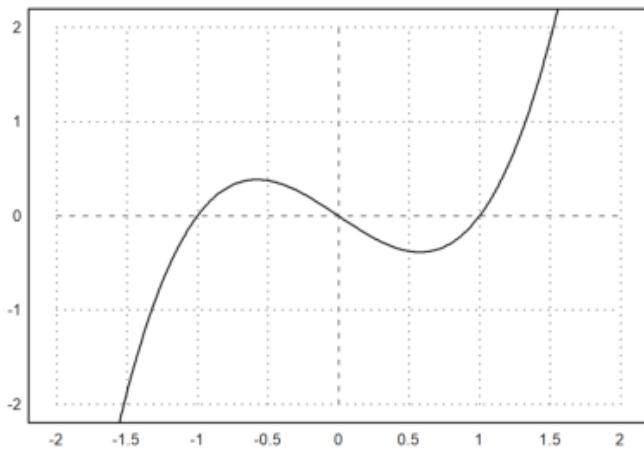
Note that x^x is not defined for $x \leq 0$. The `plot2d` function catches this error, and starts plotting as soon as the function is defined. This works for all functions which return NAN out of their range of definition.

```
>plot2d("log(x)", -0.1, 2):
```

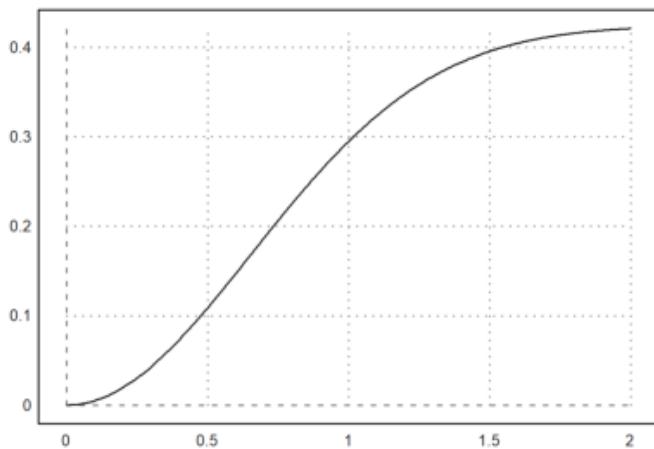


The parameter `square=true` (or `>square`) selects the y-range automatically so that the result is a square plot window. Note that by default, Euler uses a square space inside the plot window.

```
>plot2d("x^3-x", >square):
```

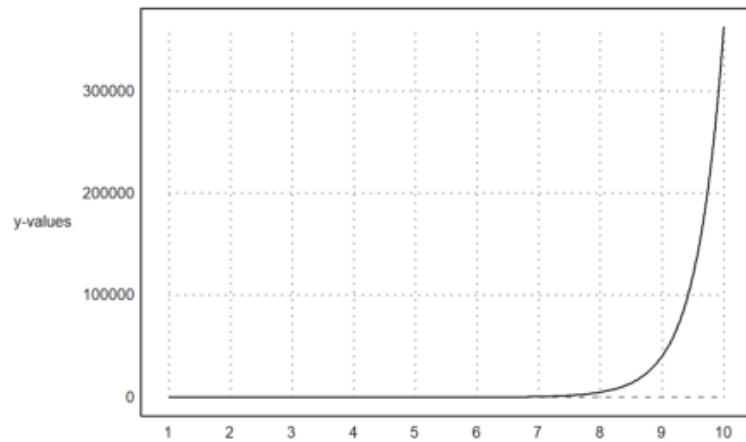


```
>plot2d(''integrate("sin(x)*exp(-x^2)", 0, x)'', 0, 2): // plot integral
```



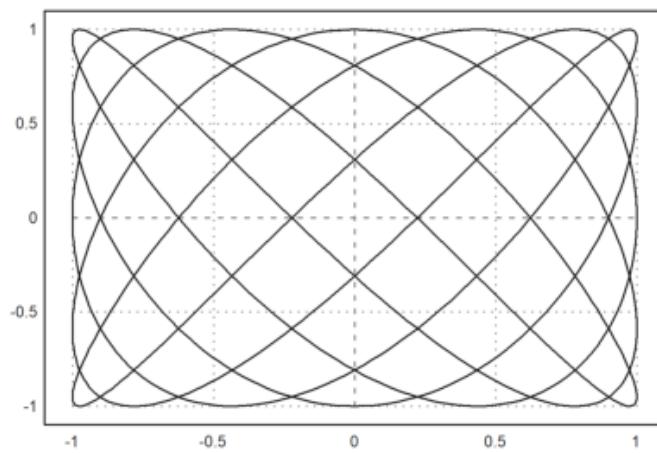
If you need more space for the y-labels, call `shrinkwindow()` with the smaller parameter, or set a positive value for "smaller" in `plot2d()`.

```
>plot2d("gamma(x)", 1, 10, yl="y-values", smaller=6, <vertical):
```

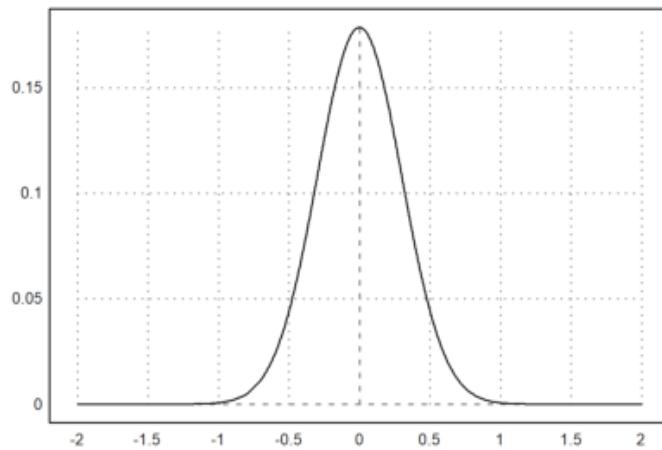


Symbolic expressions can also be used, since they are stored as simple string expressions.

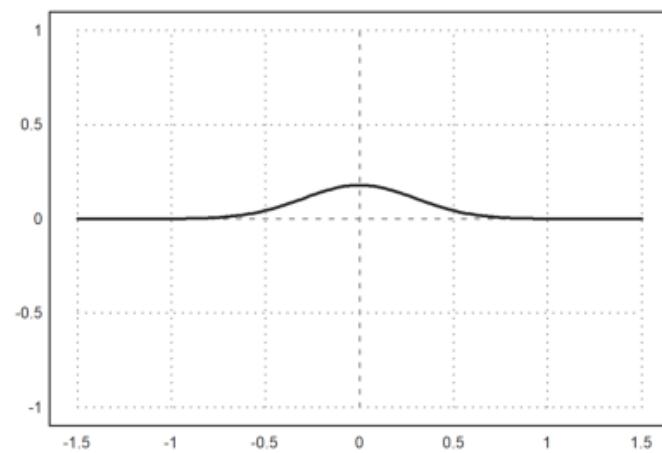
```
>x=linspace(0,2pi,1000); plot2d(sin(5x),cos(7x)):
```



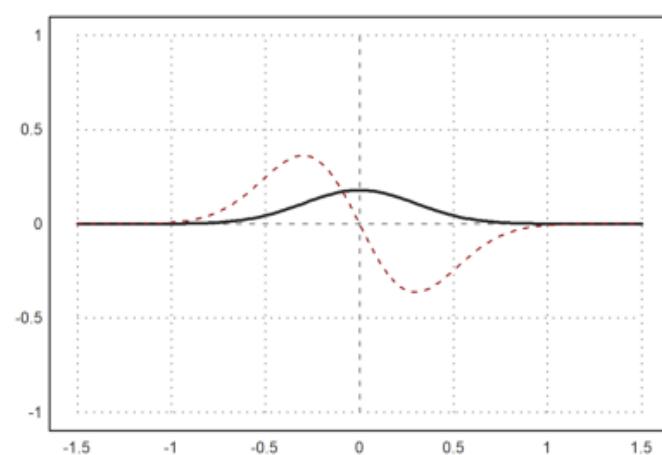
```
>a:=5.6; expr &= exp(-a*x^2)/a; // define expression
>plot2d(expr,-2,2); // plot from -2 to 2
```



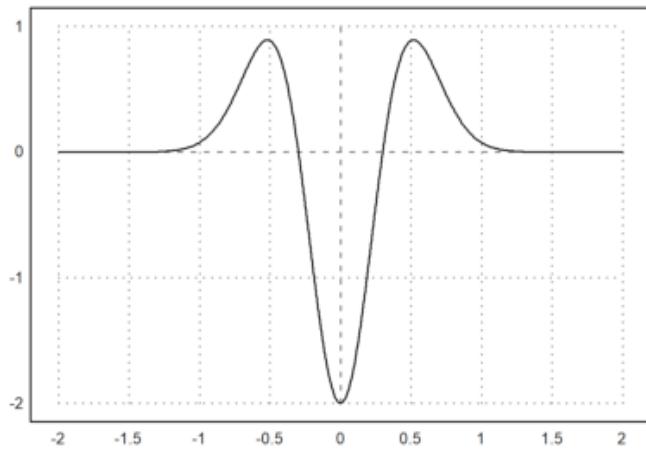
```
>plot2d(expr,r=1,thickness=2): // plot in a square around (0,0)
```



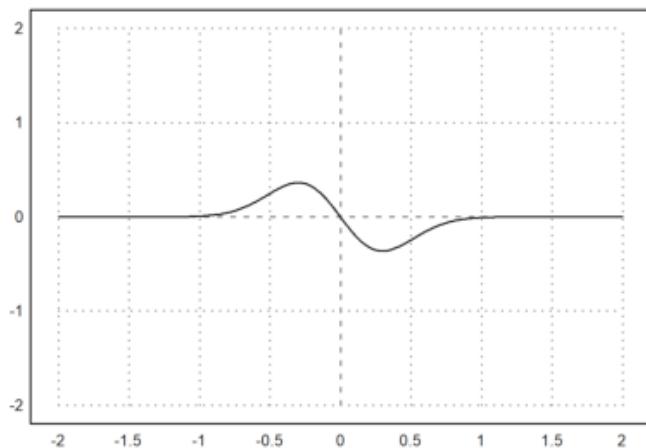
```
>plot2d(&diff(expr,x),>add,style="--",color=red): // add another plot
```



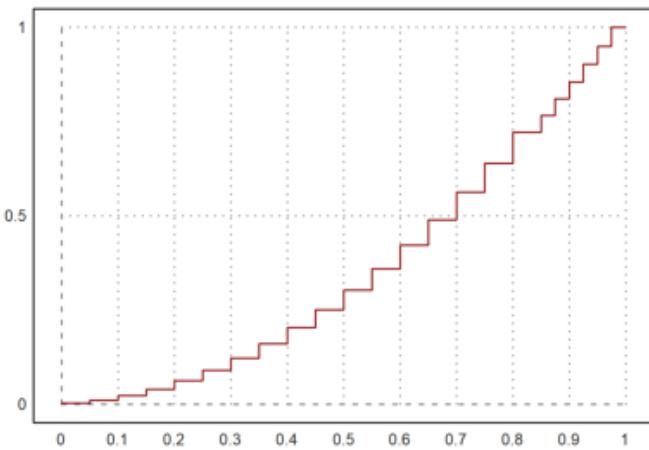
```
>plot2d(&diff(expr,x,2),a=-2,b=2,c=-2,d=1): // plot in rectangle
```



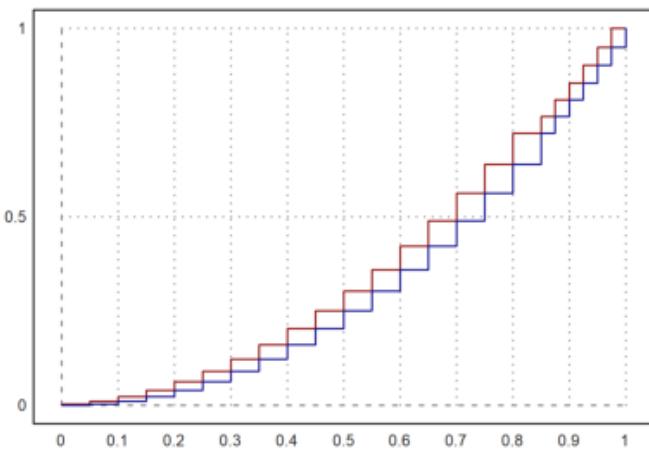
```
>plot2d(&diff(expr,x),a=-2,b=2,>square): // keep plot square
```



```
>plot2d("x^2",0,1,steps=1,color=red,n=10):
```



```
>plot2d("x^2",>add,steps=2,color=blue,n=10):
```

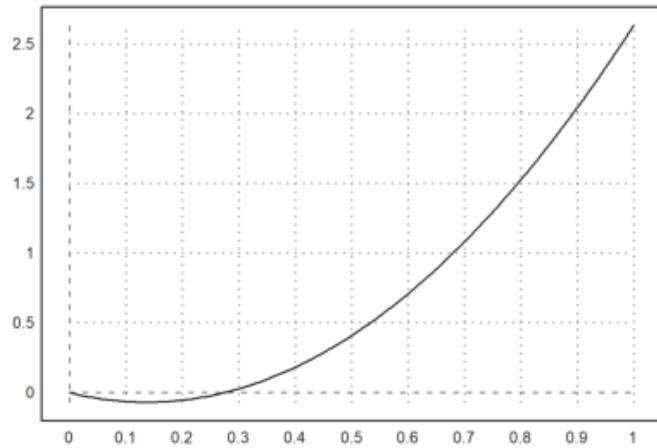


Functions in one Parameter

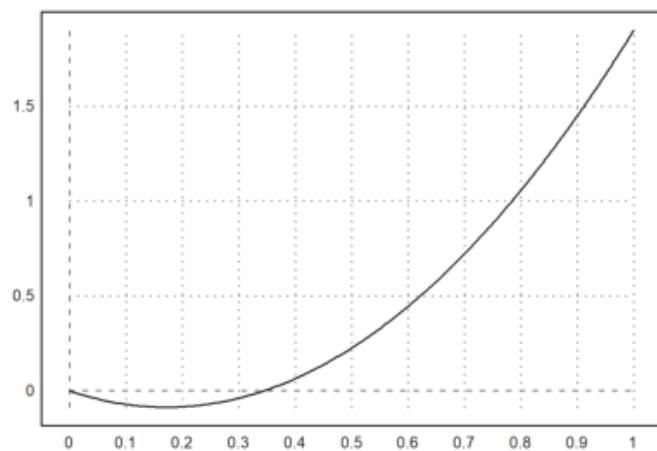
The most important plotting function for planar plots is `plot2d()`. The function is implemented in the Euler language in the file "plot.e", which is loaded at the start of the program.

Here are some examples using a function. As usual in EMT, functions that work for other functions or expressions, you can pass additional parameters (besides x) which are not global variables to the function with semicolon parameters or with a call collection.

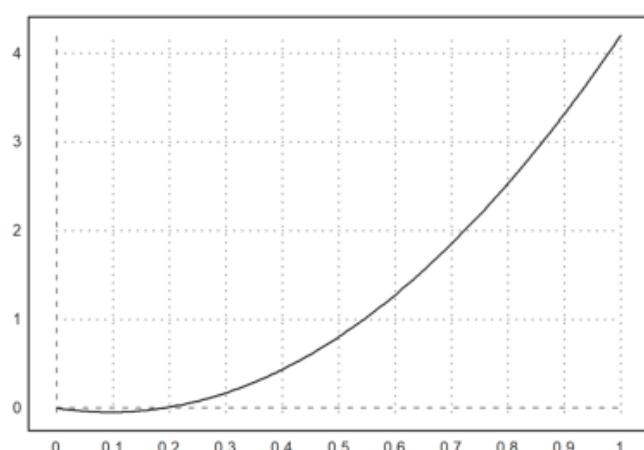
```
>function f(x,a) := x^2/a+a*x^2-x; // define a function
>a=0.3; plot2d("f",0,1;a); // plot with a=0.3
```



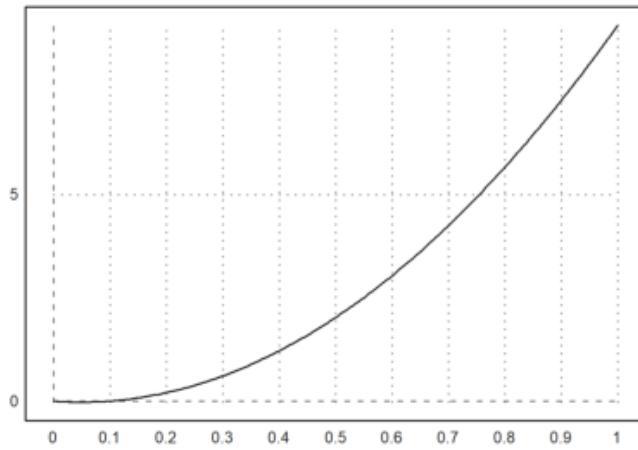
```
>plot2d("f",0,1;0.4): // plot with a=0.4
```



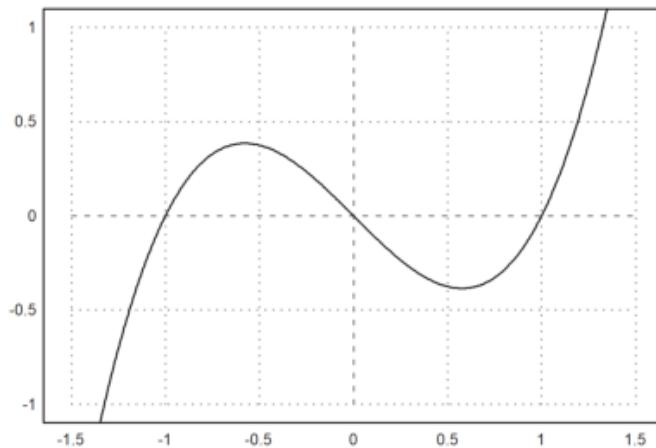
```
>plot2d({{"f",0.2}},0,1): // plot with a=0.2
```



```
>plot2d({{ "f(x,b)", b=0.1 }}, 0, 1): // plot with 0.1
```



```
>function f(x) := x^3-x; ...
>plot2d("f", r=1):
```



Here is a summary of the accepted functions

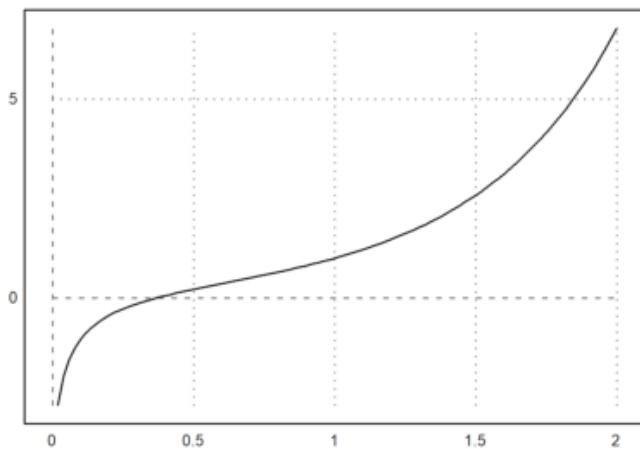
- expressions or symbolic expressions in x
- functions or symbolic functions by name as "f"
- symbolic functions just by the name f

The function `plot2d()` also accepts symbolic functions. For symbolic functions, the name alone works.

```
>function f(x) &= diff(x^x,x)
```

$$\frac{d}{dx} (x^x) = x^x (\log(x) + 1)$$

```
>plot2d(f,0,2):
```

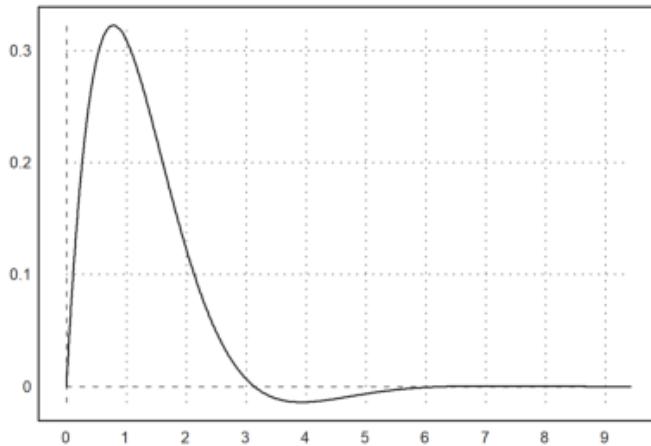


Of course, for expressions or symbolic expressions the name of the variable is enough to plot them.

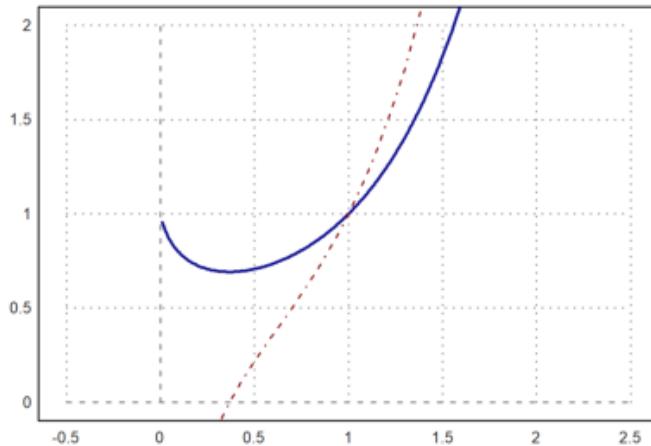
```
>expr &= sin(x)*exp(-x)
```

$$E^{-x} \sin(x)$$

```
>plot2d(expr,0,3pi):
```



```
>function f(x) &= x^x;
>plot2d(f, r=1, cx=1, cy=1, color=blue, thickness=2);
>plot2d(&diff(f(x),x), >add, color=red, style="-.-"):
```



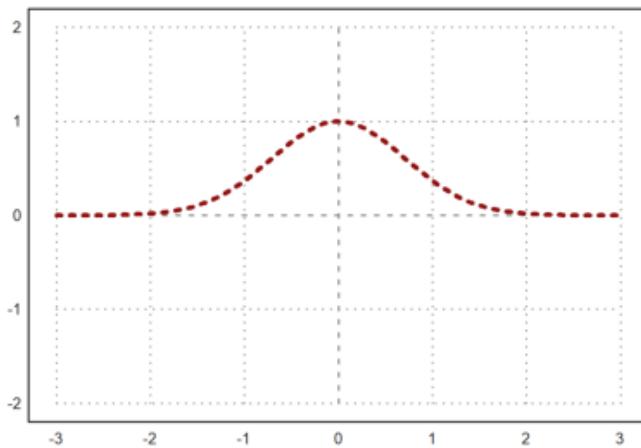
For the line style there are various options.

- style="...". Select from "-", "--", "-.", ".-", "-.-".
- color: See below for colors.
- thickness: Default is 1.

Colors can be selected as one of the default colors, or as an RGB color.

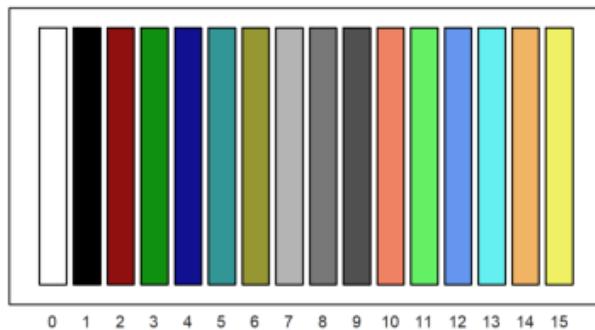
- 0..15: the default color indices.
- color constants: white, black, red, green, blue, cyan, olive, lightgray, gray, darkgray, orange, lightgreen, turquoise, lightblue, lightorange, yellow
- rgb(red,green,blue): parameters are reals in [0,1].

```
>plot2d("exp(-x^2)", r=2, color=red, thickness=3, style="--"):
```



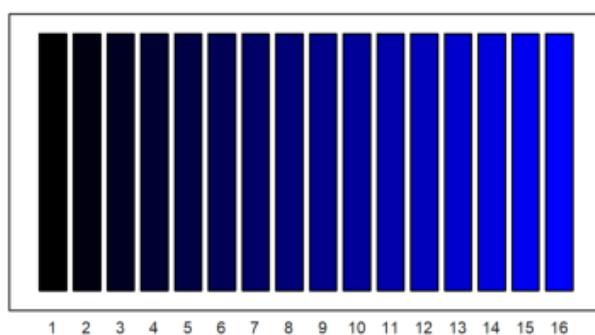
Here is a view of the predefined colors of EMT.

```
>aspect(2); columnsplot(ones(1,16), lab=0:15, grid=0, color=0:15):
```



But you can use any color.

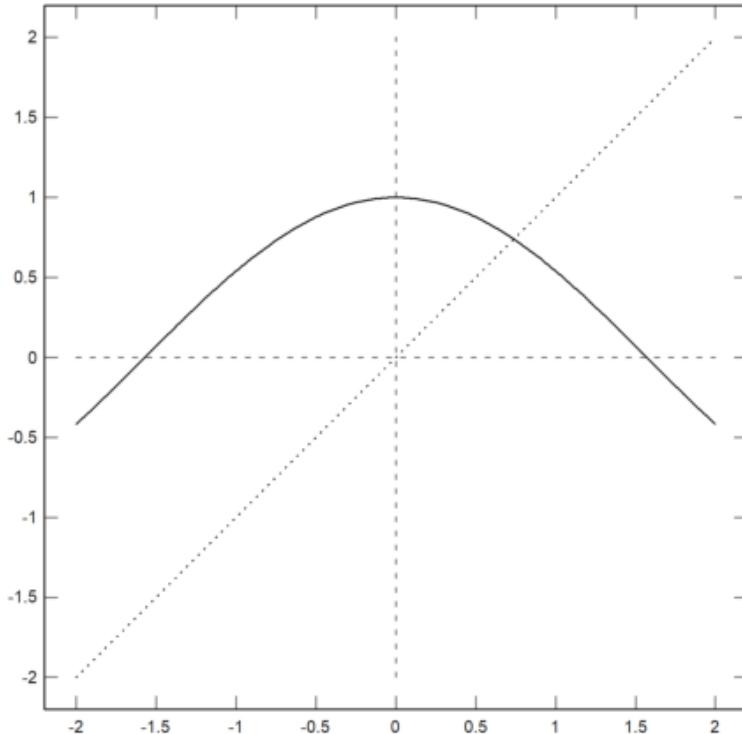
```
>columnsplot(ones(1,16), grid=0, color=rgb(0,0,linspace(0,1,15))):
```



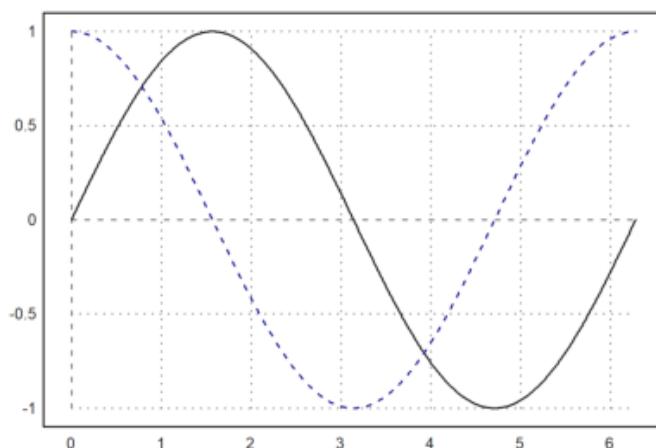
Menggambar Beberapa Kurva pada bidang koordinat yang sama

The plot more than one function (multiple functions) into one window can be done with different ways. One of the methos is using >add for several calls to plot2d in all, but the first call. We have used this feature already in the examples above.

```
>aspect(); plot2d("cos(x)",r=2,grid=6); plot2d("x",style=".",>add):
```

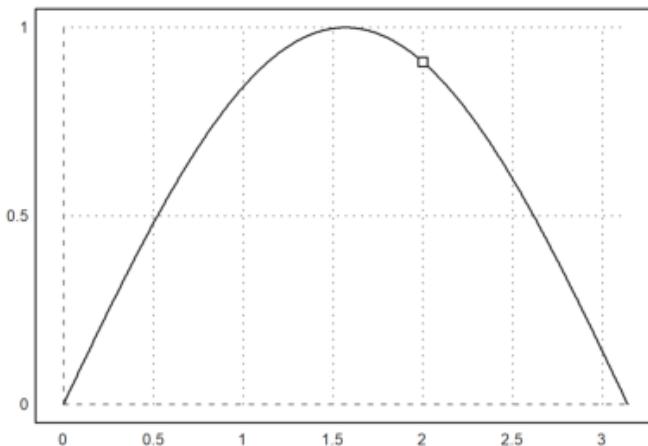


```
>aspect(1.5); plot2d("sin(x)",0,2pi); plot2d("cos(x)",color=blue,style="--":)
```



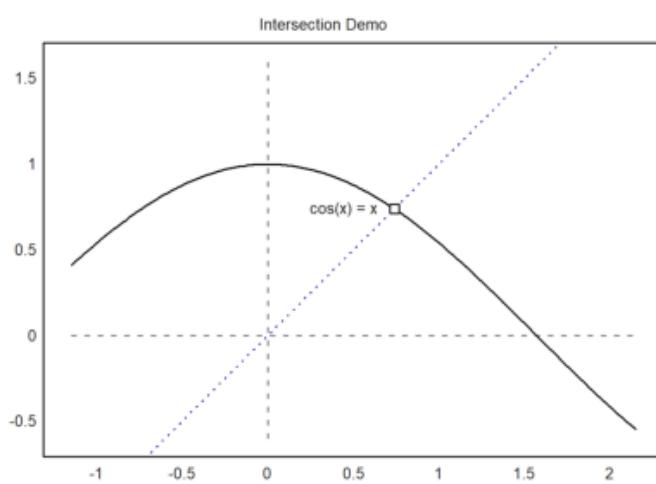
Salah satu kegunaan `>add` adalah untuk menambahkan titik pada kurva.

```
>plot2d("sin(x)",0,pi); plot2d(2,sin(2),>points,>add):
```



We add the intersection point with a label (at position "cl" for center left), and insert the result into the notebook. We also add a title to the plot.

```
>plot2d(["cos(x)", "x"], r=1.1, cx=0.5, cy=0.5, ...
> color=[black,blue], style=[ "-", "." ], ...
> grid=1);
>x0=solve("cos(x)-x",1); ...
> plot2d(x0,x0,>points,>add,title="Intersection Demo"); ...
> label("cos(x) = x",x0,x0,pos="cl",offset=20):
```

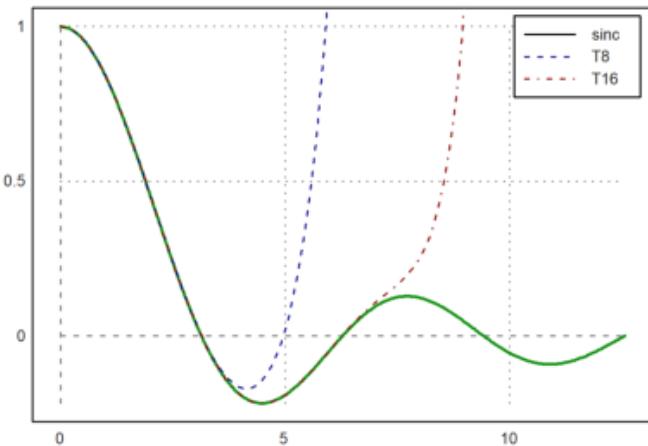


In the following demo, we plot the $\text{sinc}(x) = \sin(x)/x$ function and its 8-th and 16-th Taylor expansion. We compute this expansion using Maxima via symbolic expressions. This plot is done in the following multi-line command with three calls to `plot2d()`. The second and the third have the `>add` flag set, which makes the plots use the previous ranges. We add a label box explaining the functions.

```
>$taylor(sin(x)/x,x,0,4)
```

$$\frac{x^4}{120} - \frac{x^2}{6} + 1$$

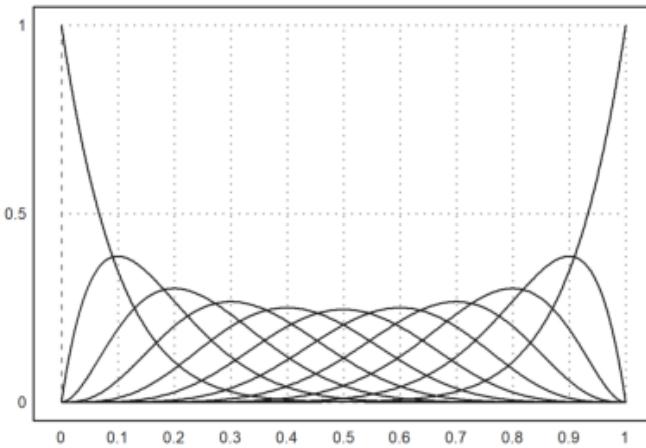
```
>plot2d("sinc(x)",0,4pi,color=green,thickness=2); ...
> plot2d(&taylor(sin(x)/x,x,0,8),>add,color=blue,style="--"); ...
> plot2d(&taylor(sin(x)/x,x,0,16),>add,color=red,style="-."); ...
> labelbox(["sinc","T8","T16"],styles=["-","--","-."], ...
> colors=[black,blue,red]):
```



In the following example, we generate the Bernstein-Polynomials.

$$B_i(x) = \binom{n}{i} x^i (1-x)^{n-i}$$

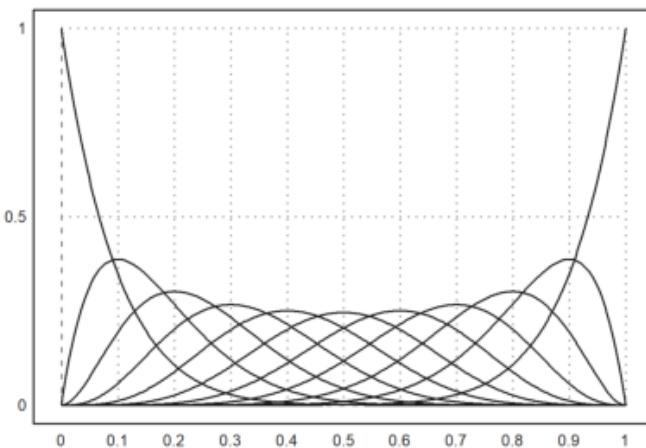
```
>plot2d("(1-x)^10",0,1); // plot first function
>for i=1 to 10; plot2d("bin(10,i)*x^i*(1-x)^(10-i)",>add); end;
>insimg;
```



The second method is using a pair of a matrix of x-values and a matrix of y-values of the same size.

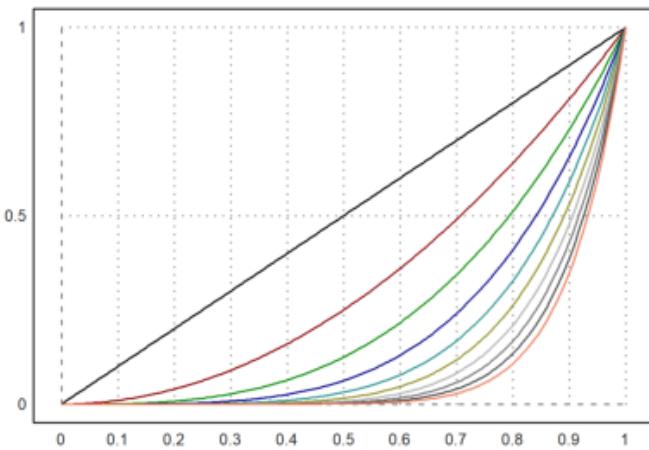
We generate a matrix of values with one Bernstein-Polynomial in each row. For this, we simply use a column vector of i. Have a look into the introduction about the matrix language to learn more details.

```
>x=linspace(0,1,500);
>n=10; k=(0:n)'; // n is row vector, k is column vector
>y=bin(n,k)*x^k*(1-x)^(n-k); // y is a matrix then
>plot2d(x,y):
```



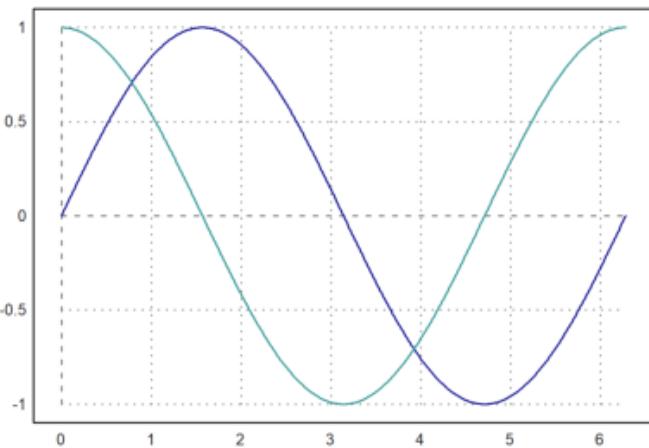
Note that the color parameter can be a vector. Then each color is used for each row of the matrix.

```
>x=linspace(0,1,200); y=x^(1:10)'; plot2d(x,y,color=1:10):
```

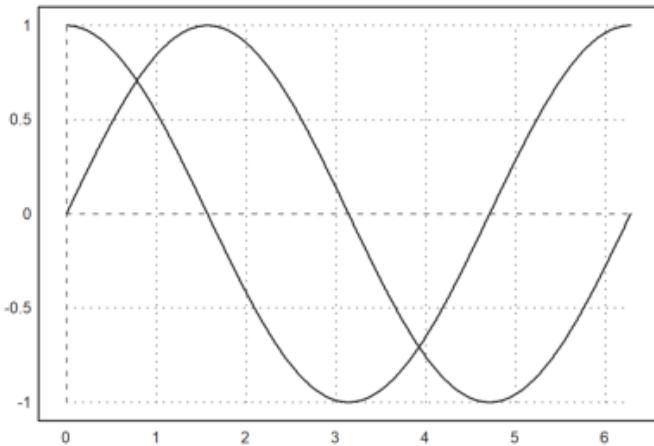


Another method is using a vector of expressions (strings). You can then use a color array, an array of styles, and an array of thicknesses of the same length.

```
>plot2d(["sin(x)", "cos(x)", 0, 2pi, color=4:5]:
```



```
>plot2d(["sin(x)", "cos(x)", 0, 2pi]: // plot vector of expressions
```



We can get such a vector from Maxima using makelist() and mxm2str().

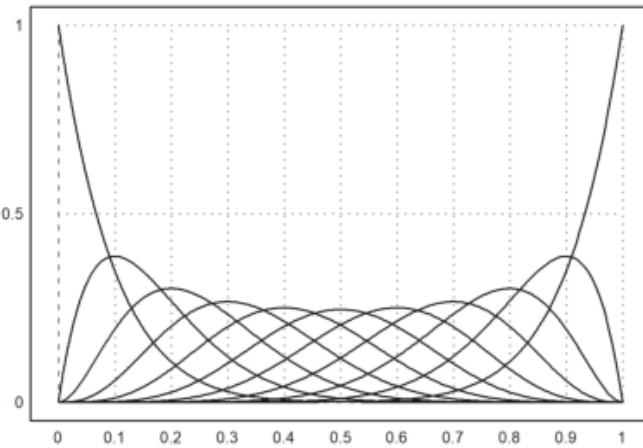
```
>v &= makelist(binomial(10,i)*x^i*(1-x)^(10-i),i,0,10) // make list
```

```
          10          9          8   2          7   3
[ (1 - x) , 10 (1 - x) x, 45 (1 - x) x , 120 (1 - x) x ,
  6   4           5   5           4   6           3   7
210 (1 - x) x , 252 (1 - x) x , 210 (1 - x) x , 120 (1 - x) x ,
  2   8           9   10
45 (1 - x) x , 10 (1 - x) x , x ]
```

```
>mxm2str(v) // get a vector of strings from the symbolic vector
```

```
(1-x)^10
10*(1-x)^9*x
45*(1-x)^8*x^2
120*(1-x)^7*x^3
210*(1-x)^6*x^4
252*(1-x)^5*x^5
210*(1-x)^4*x^6
120*(1-x)^3*x^7
45*(1-x)^2*x^8
10*(1-x)*x^9
x^10
```

```
>plot2d(mxm2str(v),0,1): // plot functions
```

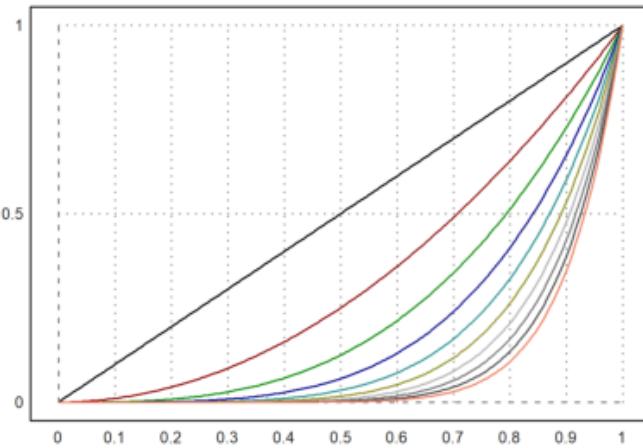


Another alternative is to use the matrix language of Euler.

If an expression produces a matrix of functions, with one function in each row, all these functions will be plotted into one plot.

For this, use a parameter vector in form of a column vector. If a color array is added it will be used for each row of the plot.

```
>n=(1:10)'; plot2d("x^n", 0, 1, color=1:10);
```

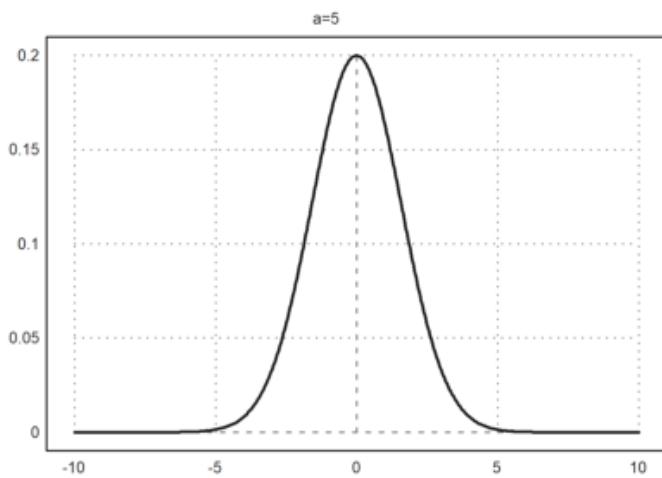


Expressions and one-line functions can see global variables.

If you cannot use a global variable, you need to use a function with an extra parameter, and pass this parameter as a semicolon parameter.

Take care, to put all assigned parameters to the end of the plot2d command. In the example we pass a=5 to the function f, which we plot from -10 to 10.

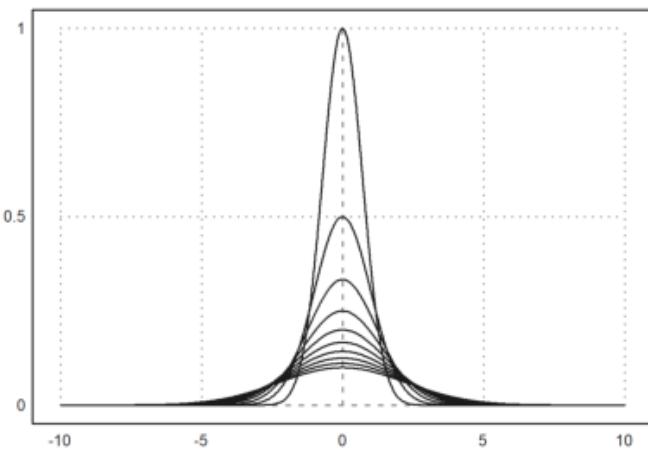
```
>function f(x,a) := 1/a*exp(-x^2/a); ...
>plot2d("f",-10,10;5,thickness=2,title="a=5"):
```



Alternatively, use a collection with the function name and all extra parameters. These special lists are called call collections, and that is the preferred way to pass arguments to a function which is itself passed as an argument to another function.

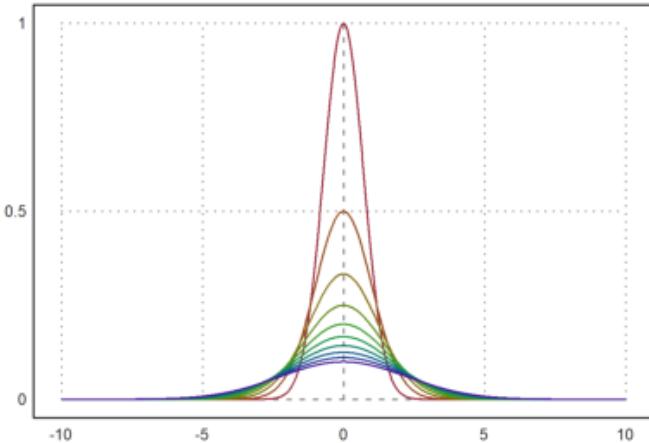
In the following example, we use a loop to plot several functions (see the tutorial about programming for loops).

```
>plot2d({{"f",1}},-10,10); ...
>for a=2:10; plot2d({{"f",a}}),>add); end:
```



We could achieve the same result in the following way using the matrix language of EMT. Each row of the matrix $f(x,a)$ is one function. Moreover, we can set colors for each row of the matrix. Double click on the function getspectral() for an explanation.

```
>x=-10:0.01:10; a=(1:10)'; plot2d(x,f(x,a),color=getspectral(a/10)):
```



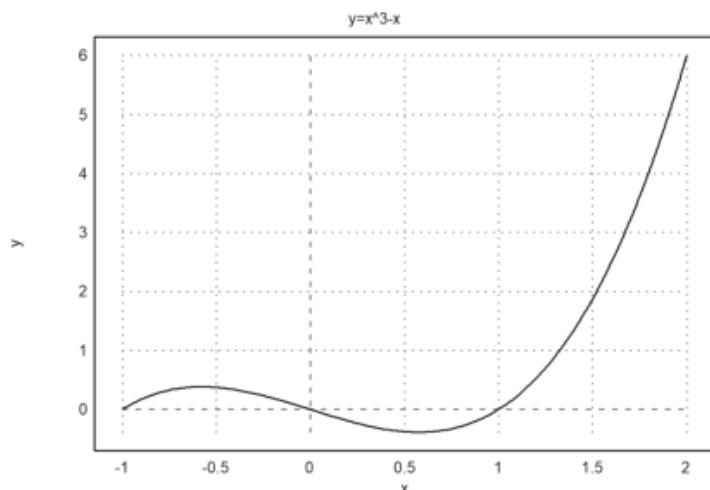
Text Labels

Simple decorations can be

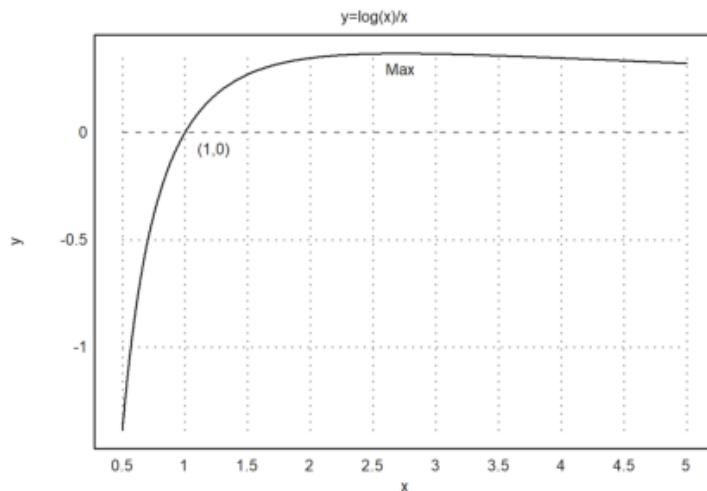
- a title with `title="..."`
- x- and y-labels with `xl="..."`, `yl="..."`
- another text label with `label("...",x,y)`

The label command will plot into the current plot at the plot oordinates (x,y). It can take a positional argument.

```
>plot2d("x^3-x",-1,2,title="y=x^3-x",yl="y",xl="x"):
```

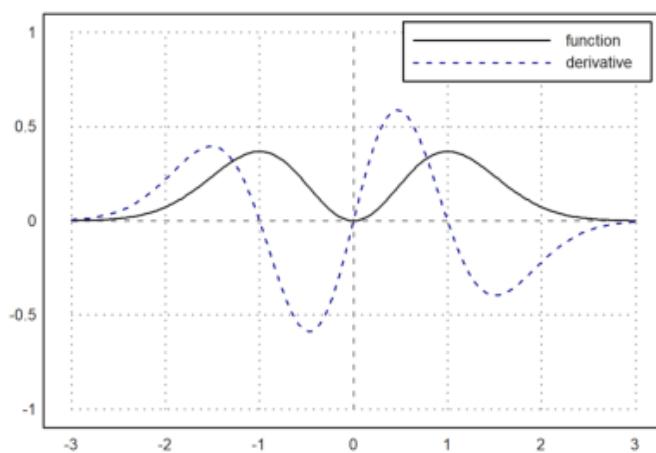


```
>expr := "log(x)/x"; ...
> plot2d(expr, 0.5, 5, title="y="+expr, xl="x", yl="y"); ...
> label("(1,0)",1,0); label("Max",E,expr(E),pos="lc"):
```



There is also the function `labelbox()`, which can show the functions and a text. It takes vectors of strings and colors, one item for each function.

```
>function f(x) &= x^2*exp(-x^2); ...
>plot2d(&f(x),a=-3,b=3,c=-1,d=1); ...
>plot2d(&diff(f(x),x),>add,color=blue,style="--"); ...
>labelbox(["function","derivative"],styles=[["-", "--"], ...
> colors=[black,blue],w=0.4):
```

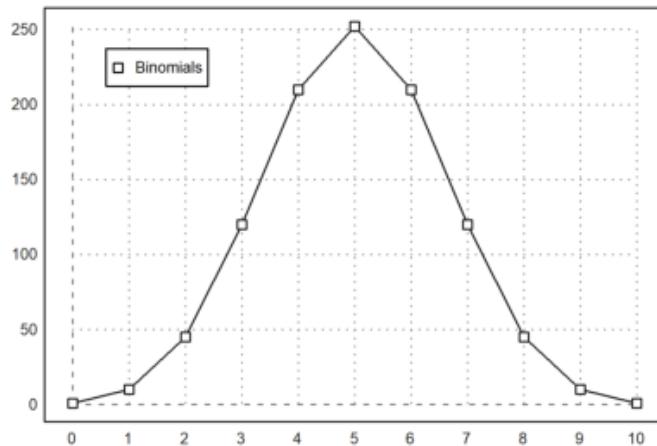


The box is anchored at the top right by default, but `>left` anchors it at the top left. You can move it to any place you like. The anchor position is the top right corner of the box, and the numbers are fractions of the size of the graphics window. The width is automatic.

For point plots, the label box works too. Add a parameter `>points`, or a vector of flags, one for each label.

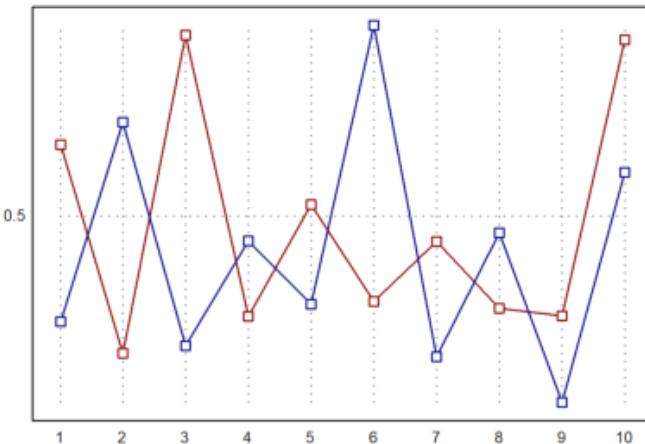
In the following example, there is only one function. So we can use strings instead of vectors of strings. We set the text color to black for this example.

```
>n=10; plot2d(0:n,bin(n,0:n),>addpoints); ...
>labelbox("Binomials",styles="[]",>points,x=0.1,y=0.1, ...
>tcolor=black,>left):
```



This style of plot is also available in `statplot()`. Like in `plot2d()` colors can be set for each row of the plot. There are more special plots for statistical purposes (see the tutorial about statistics).

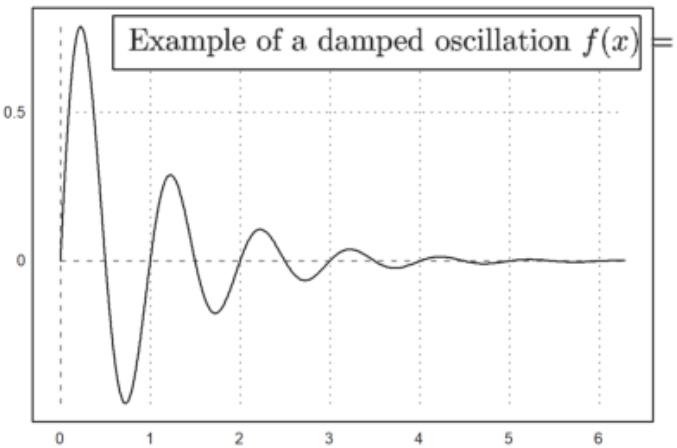
```
>statplot(1:10,random(2,10),color=[red,blue]):
```



A similar feature is the function `textbox()`.

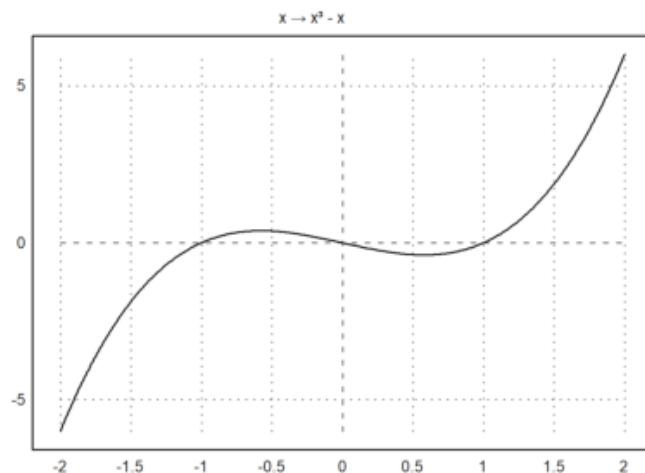
The width is by default the maximal widths of the text lines. But it can be set by the user too.

```
>function f(x) &= exp(-x)*sin(2*pi*x); ...
>plot2d("f(x)", 0, 2pi); ...
>textbox(latex("\text{Example of a damped oscillation}\\" f(x)=e^{-x}\sin(2\pi x)"));
```



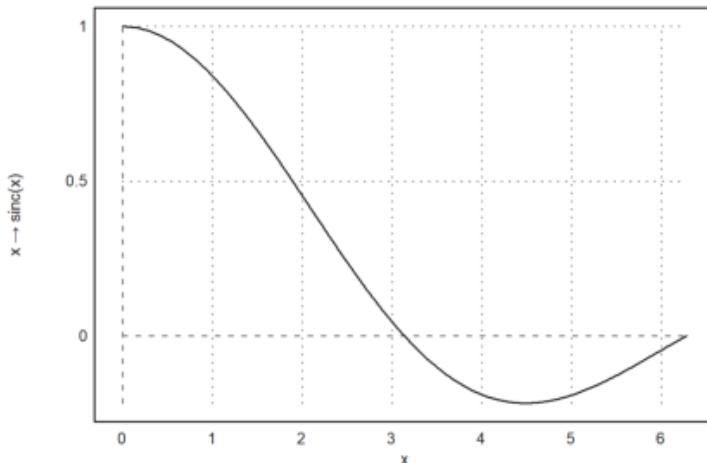
Text labels, titles, label boxes and other text can contain Unicode strings (see the syntax of EMT for more about Unicode strings).

```
>plot2d("x^3-x", title=u"x \u2192 x\u00b3 - x");
```



The labels on the x- and y-axis can be vertical, as well as the axis.

```
>plot2d("sinc(x)",0,2pi,xl="x",yl=u"x &rarr; sinc(x)",>vertical):
```



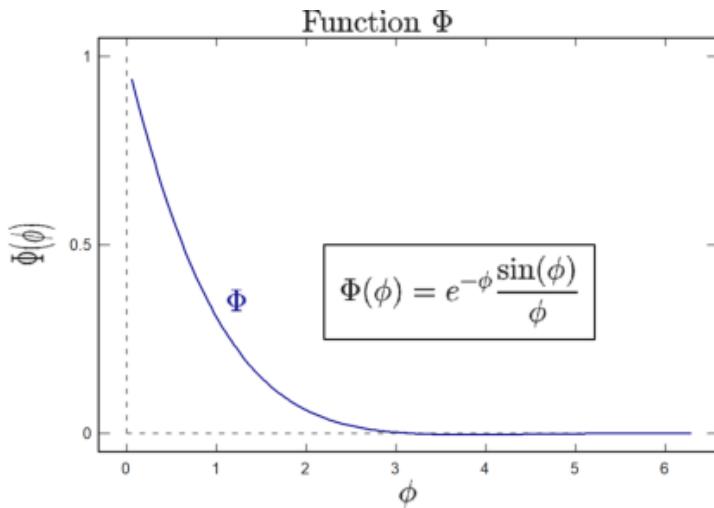
LaTeX

You can also plot LaTeX formulas if you have installed the LaTeX system. I recommend MiKTeX. The path to the binaries "latex" and "dvipng" should be in the system path, or you have to setup LaTeX in the options menu.

Note, that parsing LaTeX is slow. If you want to use LaTeX in animated plots, you should call `latex()` before the loop once and use the result (an image in a RGB matrix).

In the following plot, we use LaTeX for x- and y-labels, a label, a label box and the title of the plot.

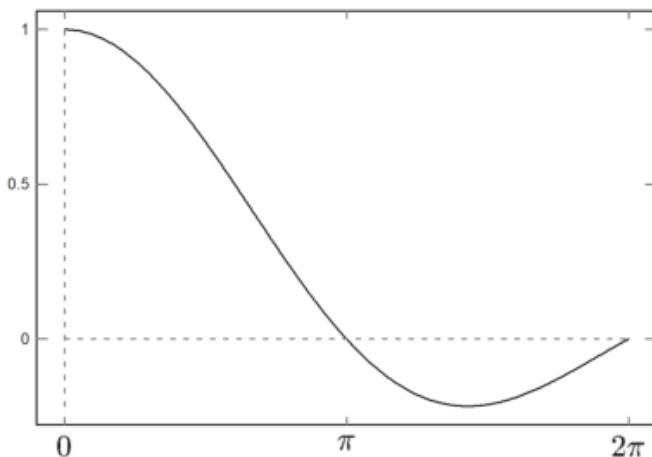
```
>plot2d("exp(-x)*sin(x)/x",a=0,b=2pi,c=0,d=1,grid=6,color=blue, ...
> title=latex("\text{Function } \$\Phi\$"), ...
> xl=latex("\phi"),yl=latex("\Phi(\phi)")); ...
>textbox( ...
> latex("\Phi(\phi) = e^{-\phi} \frac{\sin(\phi)}{\phi}"),x=0.8,y=0.5); ...
>label(latex("\Phi",color=blue),1,0.4):
```



Often, we wish a non-conformal spacing and text labels on the x-axis. We can use `xaxis()` and `yaxis()` as we will show later.

The easiest way is to do a blank plot with a frame using `grid=4`, and then to add the grids with `ygrid()` and `xgrid()`. In the following example, we use three LaTeX strings for the labels on the x-axis with `xtick()`.

```
>plot2d("sinc(x)",0,2pi,grid=4,<ticks); ...
>ygrid(-2:0.5:2,grid=6); ...
>xgrid([0:2]*pi,<ticks,grid=6); ...
>xtick([0,pi,2pi],["0","\pi","2\pi"],>latex):
```



Of course, functions can also be used.

```
>function map f(x) ...
```

```

if x>0 then return x^4
else return x^2
endif
endfunction

```

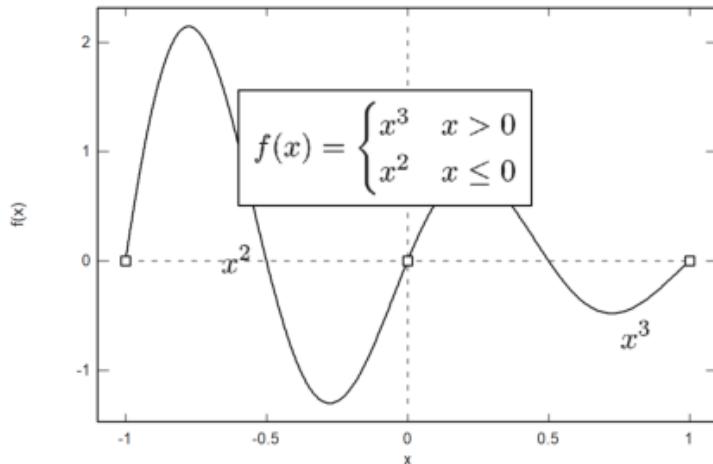
The "map" parameter helps to use the function for vectors. For the plot, it would not be necessary. But to demonstrate that vectorization is useful, we add some key points to the plot at $x=-1$, $x=0$ and $x=1$.

In the following plot, we also enter some LaTeX code. We use it for two labels and a text box. Of course, you will only be able to use LaTeX if you have LaTeX installed properly.

```

>plot2d("f",-1,1,xl="x",yl="f(x)",grid=6); ...
>plot2d([-1,0,1],f([-1,0,1]),>points,>add); ...
>label(latex("x^3"),0.72,f(0.72)); ...
>label(latex("x^2"),-0.52,f(-0.52),pos="ll"); ...
>textbox( ...
>  latex("f(x)=\begin{cases} x^3 & x>0 \\ x^2 & x \leq 0 \end{cases}"), ...
>  x=0.7,y=0.2):

```



User Interaction

When plotting a function or an expression, the parameter `>user` allows the user to zoom and shift the plot with the cursor keys or the mouse. The user can

- zoom with + or -
- move the plot with the cursor keys
- select a plot window with the mouse
- reset the view with space
- exit with return

The space key will reset the plot to the original plot window.

When plotting a data, the >user flag will simply wait for key stroke.

```
>plot2d({{"x^3-a*x"}, a=1}, >user, title="Press any key!"):
```

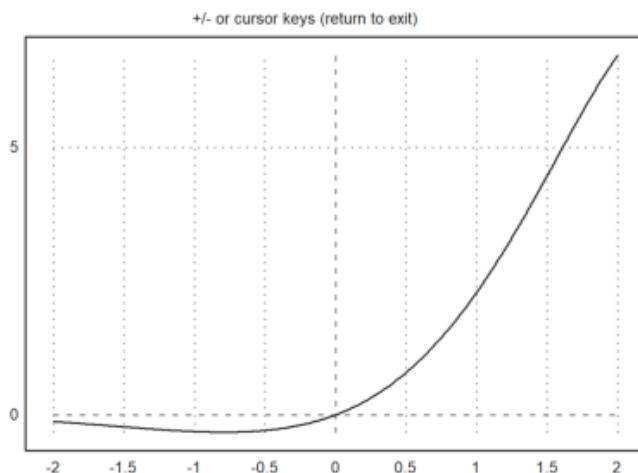
Row index 2 out of bounds!

Try "trace errors" to inspect local variables after errors.

plot2d:

```
    if auto then k2[2]=k1[2]; endif;
```

```
>plot2d("exp(x)*sin(x)", user=true, ...
> title="+/- or cursor keys (return to exit)":
```



The following demonstrates an advanced way of user interaction (see the tutorial about programming for details).

The built-in function `mousedrag()` waits for mouse or keyboard events. It reports mouse down, mouse moved or mouse up, and key presses. The function `dragpoints()` makes use of this, and lets the user drag any point in a plot.

We need a plot function first. For an example, we interpolate in 5 points with a polynomial. The function should plot into a fixed plot area.

```
>=function plotf(xp,yp,select) ...
```

Syntax error in expression, or unfinished expression!

Error in:

```
=function plotf(xp,yp,select) ... ...
^
```

```

d=interp(xp,yp);
plot2d("interpval(xp,d,x)";d,xp,r=2);
plot2d(xp,yp,>points,>add);
if select>0 then
    plot2d(xp[select],yp[select],color=red,>points,>add);
endif;
title("Drag one point, or press space or return!");
endfunction

```

Note the semicolon parameters in `plot2d` (`d` and `xp`), which are passed to the evaluation of the `interp()` function. Without this, we must write a function `plotinterp()` first, accessing the values globally.

Now we generate some random values, and let the user drag the points.

```
>t=-1:0.5:1; dragpoints("plotf",t,random(size(t))-0.5):
```

```

Variable plotf not found!
Use global variables or parameters for string evaluation.
Error in expression: plotf
Try "trace errors" to inspect local variables after errors.
dragpoints:
f$(x,y,select,args());

```

There is also a function, which plots another function depending on a vector of parameters, and lets the user adjust these parameters.

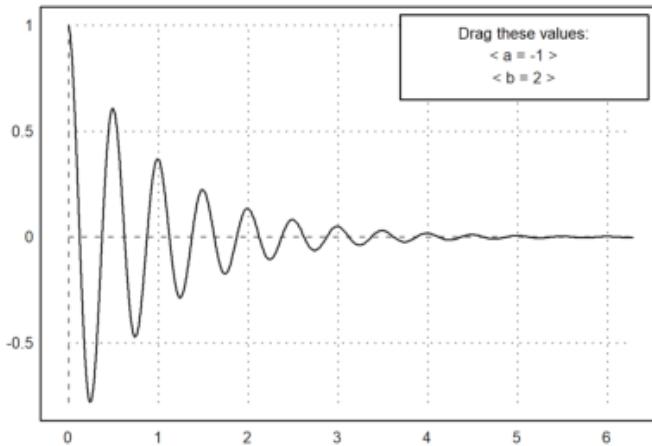
First we need the `plot` function.

```
>function plotf([a,b]) := plot2d("exp(a*x)*cos(2pi*b*x)",0,2pi;a,b);
```

Then we need names for the parameters, initial values and a $nx2$ matrix of ranges, optionally a heading line.

There are interactive sliders, which can set values by the user. The function `dragvalues()` provides this.

```
>dragvalues("plotf",["a","b"],[-1,2],[-2,2];[1,10]), ...
> heading="Drag these values:",hcolor=black):
```



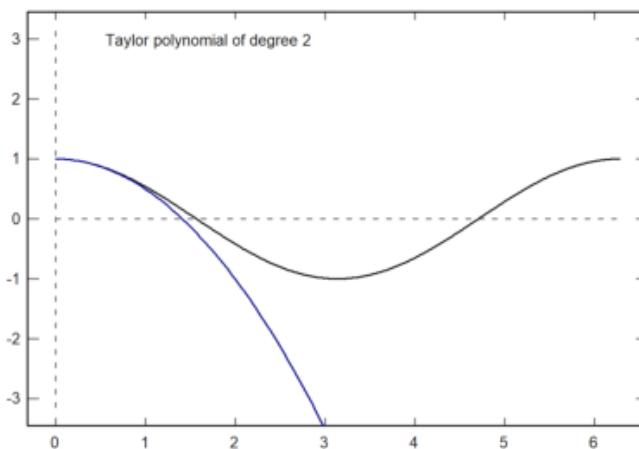
It is possible to restrict the dragged values to integers. For an example, we write a plot function, which plots a Taylor polynomial of degree n to the cosine function.

```
>function plotf(n) ...
```

```
plot2d("cos(x)", 0, 2pi, >square, grid=6);
plot2d(&"taylor(cos(x),x,0,@n)", color=blue, >add);
textbox("Taylor polynomial of degree "+n, 0.1, 0.02, style="t", >left);
endfunction
```

Now we allow the degree n to vary from 0 to 20 in 20 stops. The result of dragvalues() is used to plot the sketch with this n , and to insert the plot into the notebook.

```
>nd=dragvalues("plotf", "degree", 2, [0,20], 20, y=0.8, ...
> heading="Drag the value:"); ...
>plotf(nd);
```



The following is a simple demonstration of the function. The user can draw over the plot window, leaving a trace of points.

```
>function dragtest ...
```

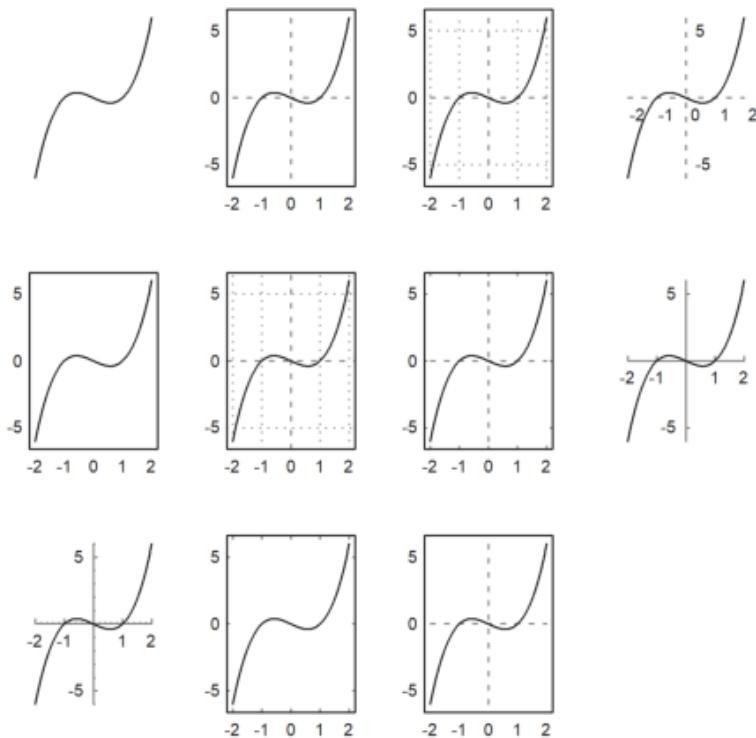
```
plot2d(none,r=1,title="Drag with the mouse, or press any key!");
start=0;
repeat
  {flag,m,time}=mousedrag();
  if flag==0 then return; endif;
  if flag==2 then
    hold on; mark(m[1],m[2]); hold off;
  endif;
end
endfunction
```

```
>dragtest // lihat hasilnya dan cobalah lakukan!
```

Styles of 2D Plots

By default, EMT computes automatic axis ticks and adds labels to each tick. This can be changed with the grid parameter. The default style of the axis and the labels can be modified. Additionally, labels and a title can be added manually. To reset to the default styles, use reset().

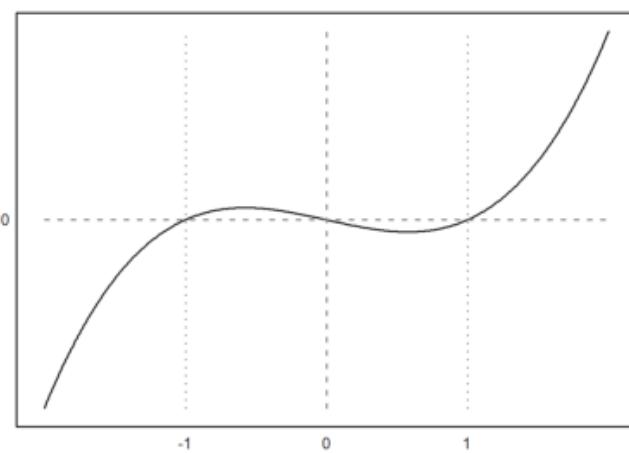
```
>aspect();
>figure(3,4); ...
> figure(1); plot2d("x^3-x",grid=0); ... // no grid, frame or axis
> figure(2); plot2d("x^3-x",grid=1); ... // x-y-axis
> figure(3); plot2d("x^3-x",grid=2); ... // default ticks
> figure(4); plot2d("x^3-x",grid=3); ... // x-y- axis with labels inside
> figure(5); plot2d("x^3-x",grid=4); ... // no ticks, only labels
> figure(6); plot2d("x^3-x",grid=5); ... // default, but no margin
> figure(7); plot2d("x^3-x",grid=6); ... // axes only
> figure(8); plot2d("x^3-x",grid=7); ... // axes only, ticks at axis
> figure(9); plot2d("x^3-x",grid=8); ... // axes only, finer ticks at axis
> figure(10); plot2d("x^3-x",grid=9); ... // default, small ticks inside
> figure(11); plot2d("x^3-x",grid=10); ...// no ticks, axes only
> figure(0):
```



The parameter `<frame` switches off the frame, and `framecolor=blue` sets the frame to a blue color.

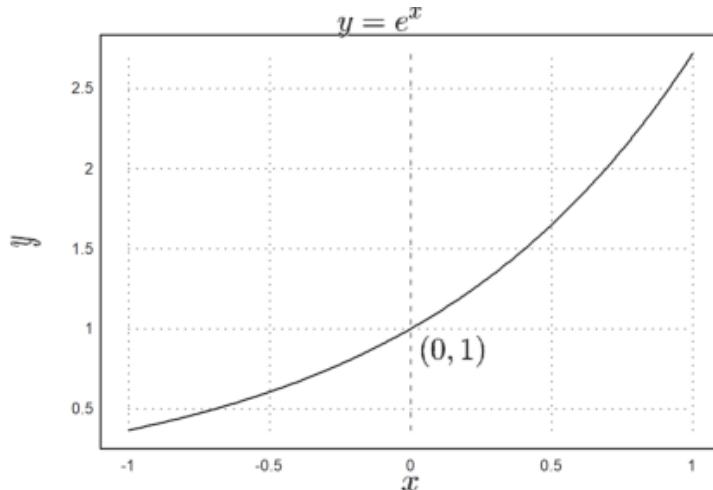
If you want your own ticks, you can use `style=0`, and add everything later.

```
>aspect(1.5);
>plot2d("x^3-x",grid=0); // plot
>frame; xgrid([-1,0,1]); ygrid(0); // add frame and grid
```



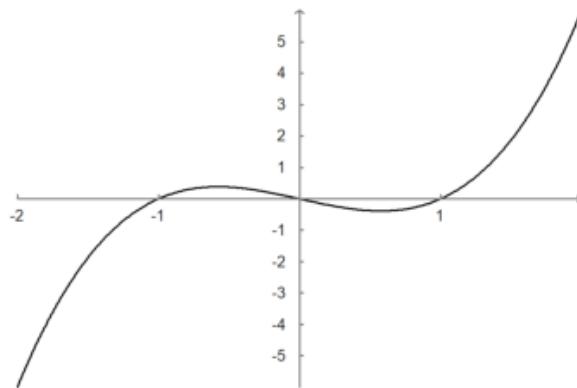
For the plot title and labels of the axes, see the following example.

```
>plot2d("exp(x)",-1,1);
>textcolor(black); // set the text color to black
>title(latex("y=e^x")); // title above the plot
>xlabel(latex("x")); // "x" for x-axis
>ylabel(latex("y"),>vertical); // vertical "y" for y-axis
>label(latex("(0,1)",0,1,color=blue): // label a point
```



The axis can be drawn separately with `xaxis()` and `yaxis()`.

```
>plot2d("x^3-x",<grid,<frame);
>xaxis(0,xx=-2:1,style="->"); yaxis(0,yy=-5:5,style="->"):
```

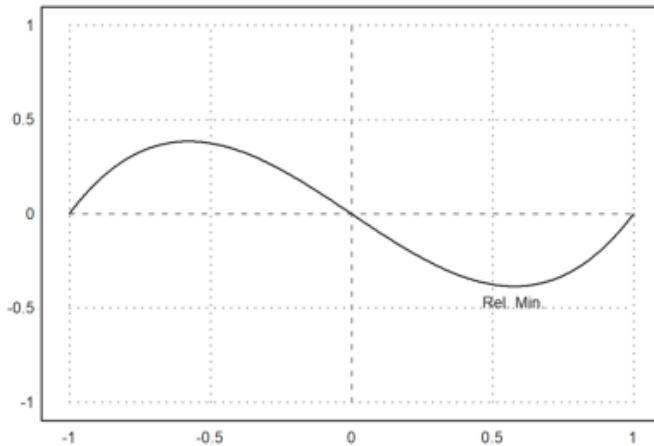


Text on the plot can be set with `label()`. In the following example, "lc" means lower center. It sets the position of the label relative to the plot coordinates.

```
>function f(x) &= x^3-x
```

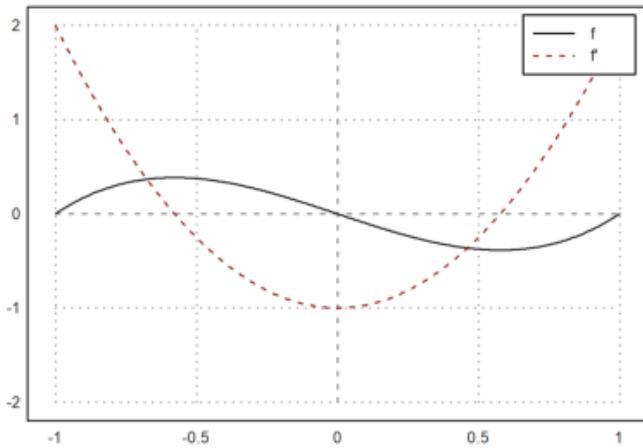
$$x^3 - x$$

```
>plot2d(f,-1,1,>square);
>x0=fmin(f,0,1); // compute point of minimum
>label("Rel. Min.",x0,f(x0),pos="lc"); // add a label there
```

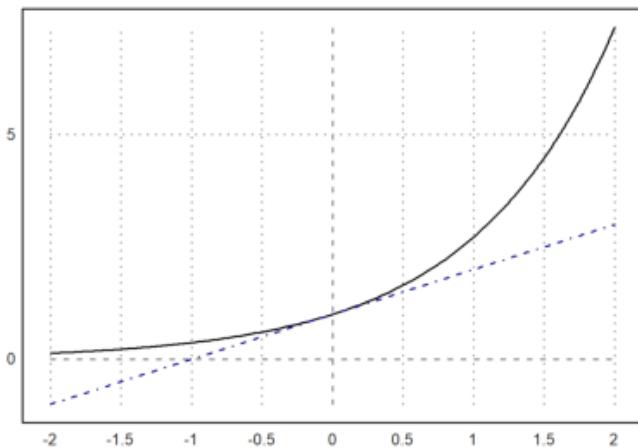


There are also text boxes.

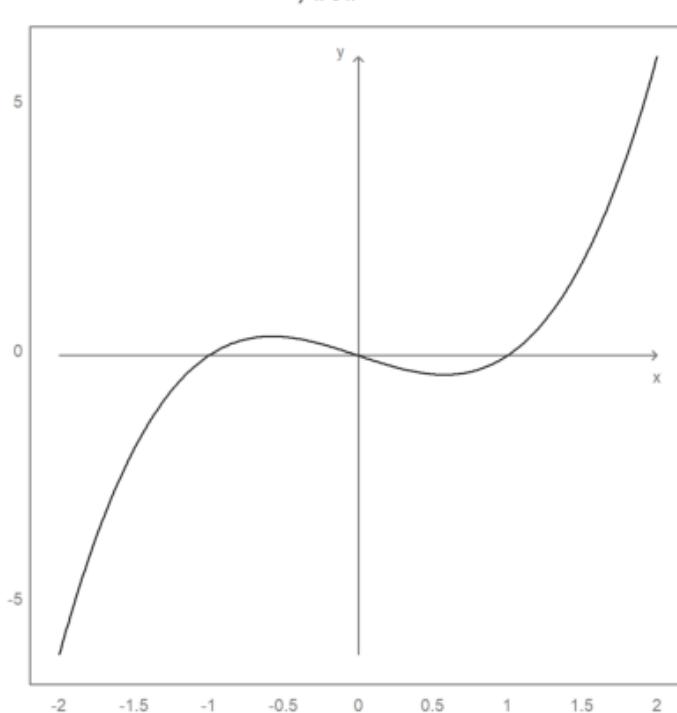
```
>plot2d(&f(x),-1,1,-2,2); // function
>plot2d(&diff(f(x),x),>add,style="--",color=red); // derivative
>labelbox(["f","f'"],["-", "--"],[black,red]): // label box
```



```
>plot2d(["exp(x)", "1+x"], color=[black, blue], style=["-", "-.-"]):
```



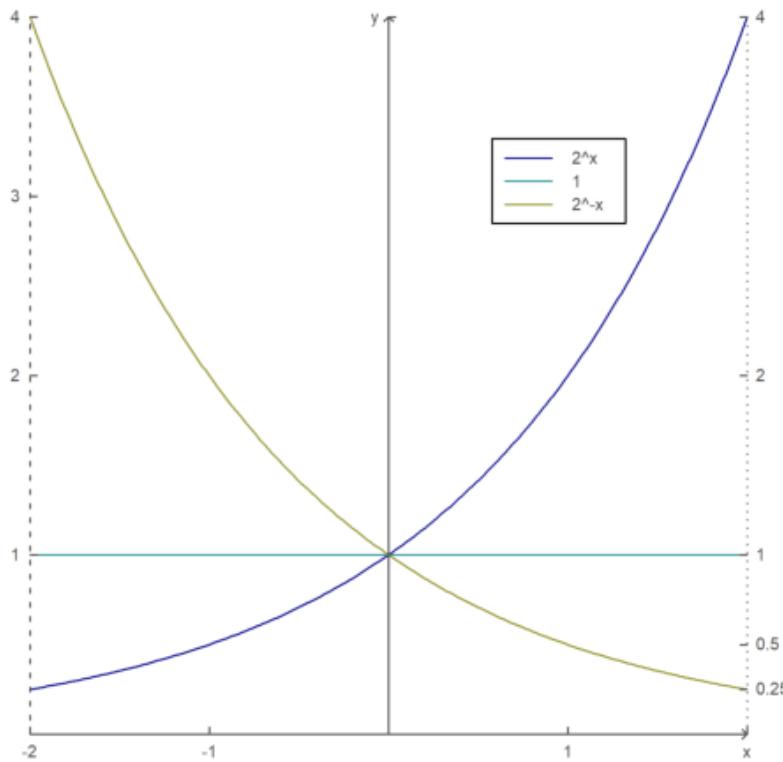
```
>gridstyle("->", color=gray, textcolor=gray, framecolor=gray); ...
> plot2d("x^3-x", grid=1); ...
> setttitle("y=x^3-x", color=black); ...
> label("x", 2, 0, pos="bc", color=gray); ...
> label("y", 0, 6, pos="cl", color=gray); ...
> reset():
```



For even more control, the x-axis and the y-axis can be done manually.

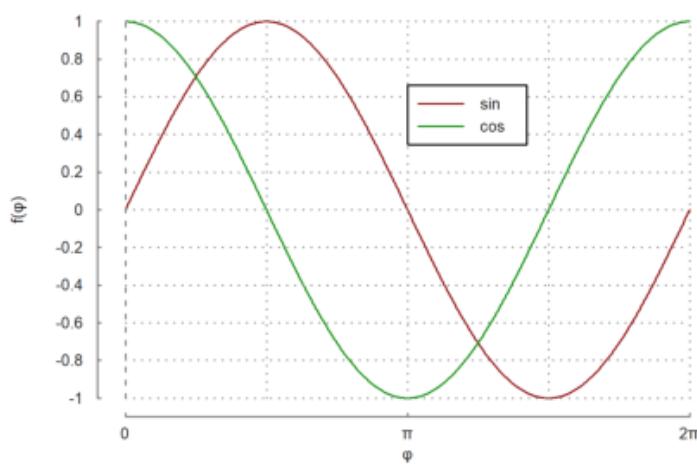
The command `fullwindow()` expands the plot window since we no longer need place for labels outside the plot window. Use `shrinkwindow()` or `reset()` to reset to the defaults.

```
>fullwindow; ...
> gridstyle(color=darkgray,textcolor=darkgray); ...
> plot2d(["2^x","1","2^(-x)"],a=-2,b=2,c=0,d=4,<grid,color=4:6,<frame); ...
> xaxis(0,-2:1,style="->"); xaxis(0,2,"x",<axis); ...
> yaxis(0,4,"y",style="->"); ...
> yaxis(-2,1:4,>left); ...
> yaxis(2,2^(-2:2),style=".",<left); ...
> labelbox(["2^x","1","2^-x"],colors=4:6,x=0.8,y=0.2); ...
> reset:
```



Here is another example, where Unicode strings are used and axes outside the plot area.

```
>aspect(1.5);
>plot2d(["sin(x)", "cos(x")], 0, 2pi, color=[red, green], <grid, <frame); ...
> xaxis(-1.1, (0:2)*pi, xt=["0", u"\u03c0", "u"2\pi"], style="-", >ticks, >zero);
> xgrid((0:0.5:2)*pi, <ticks); ...
> yaxis(-0.1*pi, -1:0.2:1, style="-", >zero, >grid); ...
> labelbox(["sin", "cos"], colors=[red, green], x=0.5, y=0.2, >left); ...
> xlabel(u"\u03c6"); ylabel(u"f(\u03c6)"):
```



Plotting 2D Data

If x and y are data vectors, these data will be used as x - and y -coordinates of a curve. In this case, a , b , c , and d , or a radius r can be specified, or the plot window will adjust automatically to the data. Alternatively, `>square` can be set to keep a square aspect ratio.

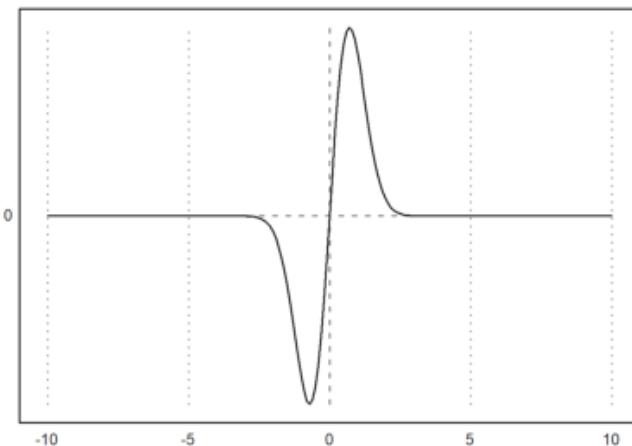
Plotting an expression is only an abbreviation for data plots. For data plots, you need one or more rows of x -values, and one or more rows of y -values. From the range and the x -values the `plot2d` function will compute the data to plot, by default with adaptive evaluation of the function. For point plots use "`>points`", for mixed lines and points use "`>addpoints`".

But you can enter data directly.

- Use row vectors for x and y for one function.
- Matrices for x and y are plotted line by line.

Here is an example with one row for x and y .

```
>x=-10:0.1:10; y=exp(-x^2)*x; plot2d(x,y):
```



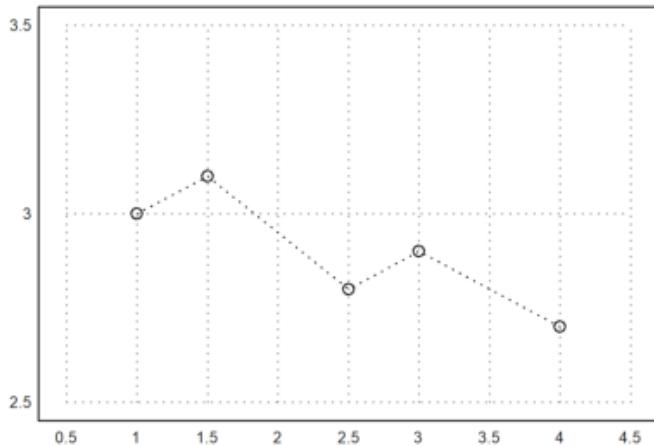
Data can also be plotted as points. Use `points=true` for this. The plot works like polygons, but draws only the corners.

- `style="...":` Select from "[", "<>", "o", ".", "..", "+", "*", "[", "<>", "o", "..", "", "|".

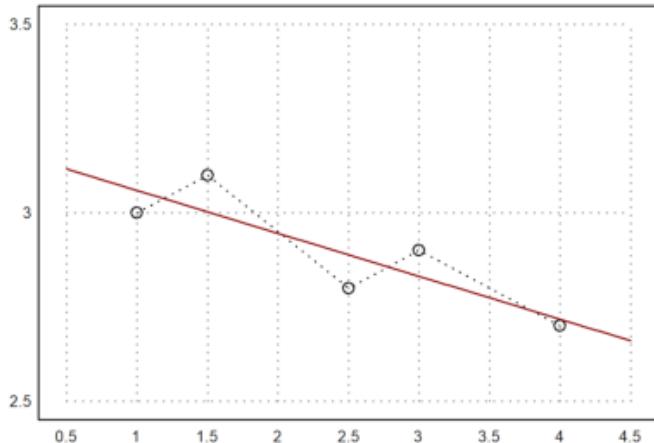
To plot sets of points use `>points`. If the color is a vector of colors, each points gets a different color. For a matrix of coordinates and a column vector, the color applies to the rows of the matrix.

The parameter `>addpoints` adds points to line segments for plots of data.

```
>xdata=[1,1.5,2.5,3,4]; ydata=[3,3.1,2.8,2.9,2.7]; // data
>plot2d(xdata,ydata,a=0.5,b=4.5,c=2.5,d=3.5,style="."); // lines
>plot2d(xdata,ydata,>points,>add,style="o"): // add points
```



```
>p=polyfit(xdata,ydata,1); // get regression line
>plot2d("polyval(p,x)",>add,color=red); // add plot of line
```



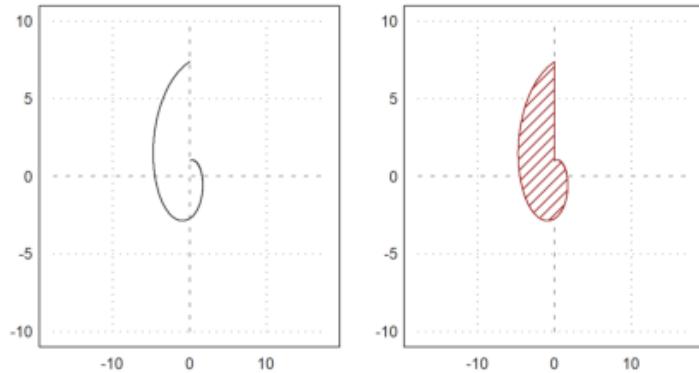
Menggambar Daerah Yang Dibatasi Kurva

Data plots are really polygons. We can also plot curves or filled curves.

- filled=true fills the plot.
- style="...": Select from "", "/", "\", "\\".
- fillcolor: See above for available colors.

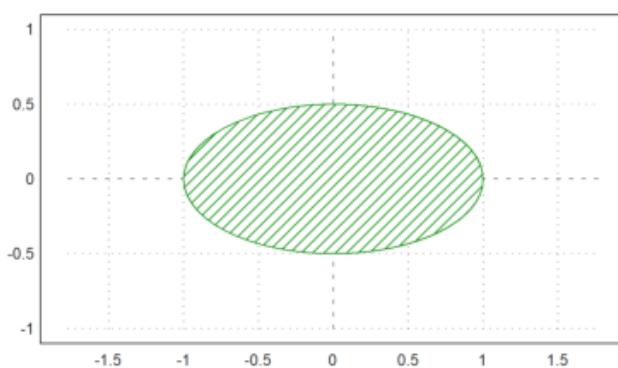
The fill color is determined by the argument "fillcolor", and an optional <outline> prevents drawing the boundary for all styles but the default one.

```
>t=linspace(0,2pi,1000); // parameter for curve
>x=sin(t)*exp(t/pi); y=cos(t)*exp(t/pi); // x(t) and y(t)
>figure(1,2); aspect(16/9)
>figure(1); plot2d(x,y,r=10); // plot curve
>figure(2); plot2d(x,y,r=10,>filled,style="/",fillcolor=red); // fill curve
>figure(0):
```

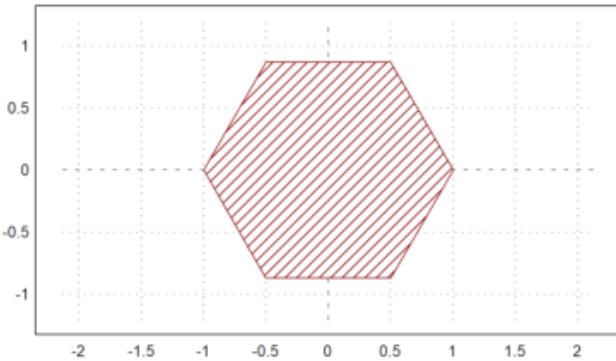


In the following examples we plot a filled ellips and two filled hexagons using a closed curve with 6 points with different fill style.

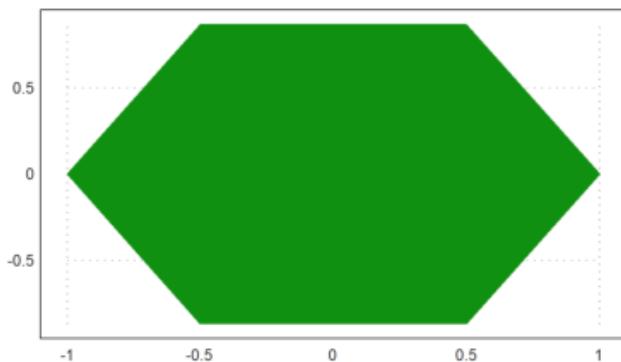
```
>x=linspace(0,2pi,1000); plot2d(sin(x),cos(x)*0.5,r=1,>filled,style="/"):
```



```
>t=linspace(0,2pi,6); ...
>plot2d(cos(t),sin(t),>filled,style="/",fillcolor=red,r=1.2):
```

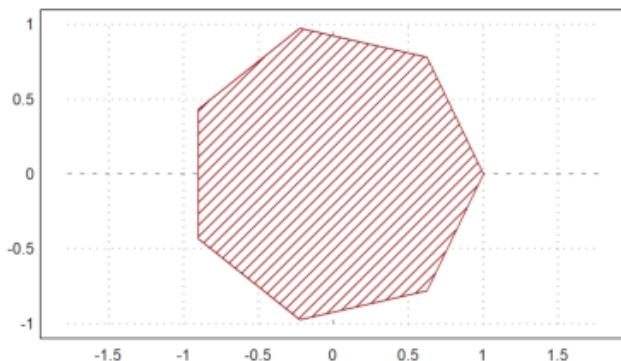


```
>t=linspace(0,2pi,6); plot2d(cos(t),sin(t),>filled,style="#"):
```



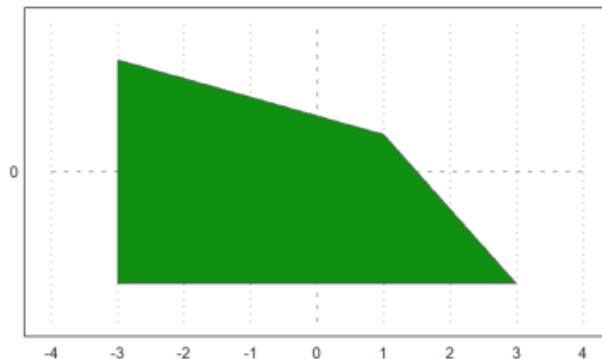
Another example is a septagon, which we create with 7 points on the unit circle.

```
>t=linspace(0,2pi,7); ...
> plot2d(cos(t),sin(t),r=1,>filled,style="/",fillcolor=red):
```



The following is the set of the maximal value of four linear conditions less than or equal 3. This is $A[k].v \leq 3$ for all rows of A . To get nice corners, we use n relatively large.

```
>A=[2,1;1,2;-1,0;0,-1];
>function f(x,y) := max([x,y].A');
>plot2d("f",r=4,level=[0;3],color=green,n=111):
```

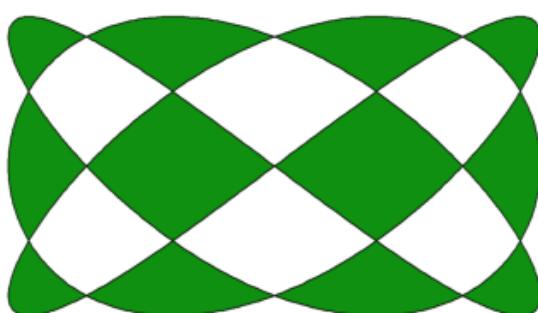


The main point of the matrix language is that it allows to generate tables of functions easily.

```
>t=linspace(0,2pi,1000); x=cos(3*t); y=sin(4*t);
```

We now have vectors x and y of values. `plot2d()` can plot these values as a curve connecting the points. The plot can be filled. In this case this yields a nice result due to the winding rule, which is used for the fill.

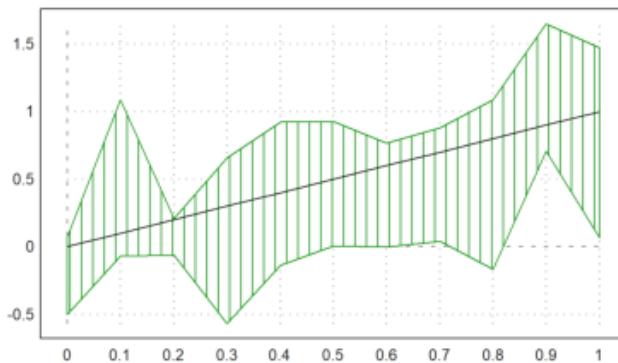
```
>plot2d(x,y,<grid,<frame,>filled):
```



A vector of intervals is plotted against x values as a filled region between lower and upper values of the intervals.

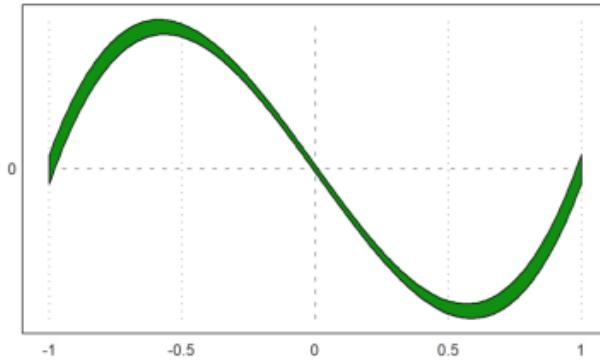
This is can be useful to plot errors of the computation. But it can also be used to plot statistical errors.

```
>t=0:0.1:1; ...
> plot2d(t,interval(t-random(size(t)),t+random(size(t))),style="|");
> plot2d(t,t,add=true); ...
```



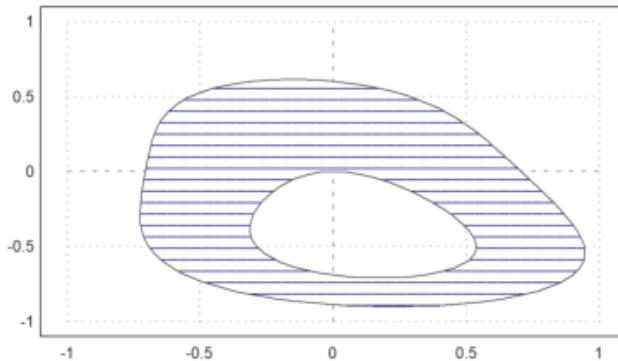
If x is a sorted vector, and y is a vector of intervals, then plot2d will plot the filled ranges of the intervals in the plane. The fill styles are the same as the styles of polygons.

```
>t=-1:0.01:1; x=~t-0.01,t+0.01~; y=x^3-x;
>plot2d(t,y);
```



It is possible to fill regions of values for a specific function. For this, level must be a 2xn matrix. The first row are the lower bounds and the second row contains the upper bounds.

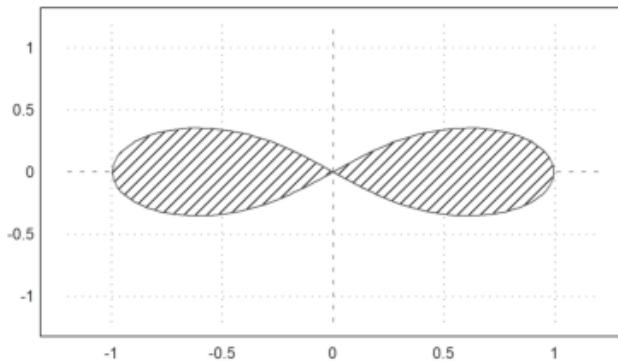
```
>expr := "2*x^2+x*y+3*y^4+y"; // define an expression f(x,y)
>plot2d(expr,level=[0;1],style="-",color=blue): // 0 <= f(x,y) <= 1
```



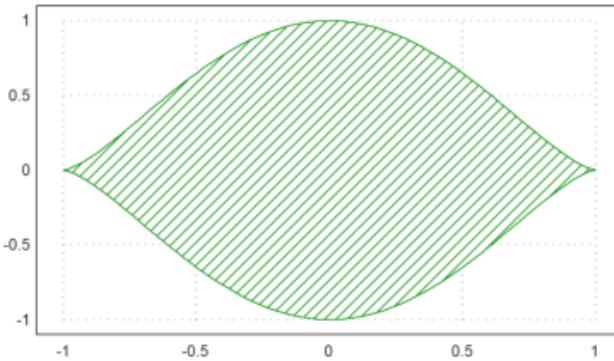
We can also fill ranges of values like

$$-1 \leq (x^2 + y^2)^2 - x^2 + y^2 \leq 0.$$

```
>plot2d("(x^2+y^2)^2-x^2+y^2",r=1..2,level=[-1;0],style="/"):
```



```
>plot2d("cos(x)","sin(x)^3",xmin=0,xmax=2pi,>filled,style="/"):
```



Grafik Fungsi Parametrik

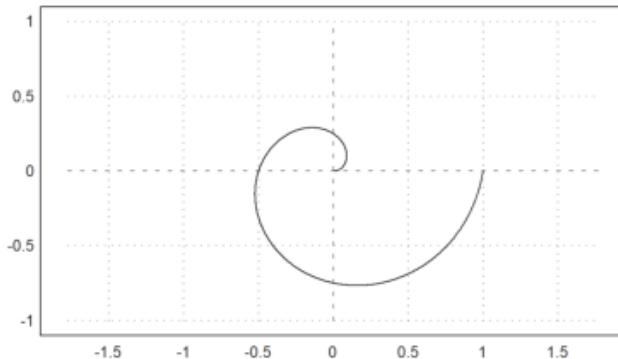
The x-values need not be sorted. (x,y) simply describes a curve. If x is sorted, the curve is a graph of a function.

In the following example, we plot the spiral

$$\gamma(t) = t \cdot (\cos(2\pi t), \sin(2\pi t))$$

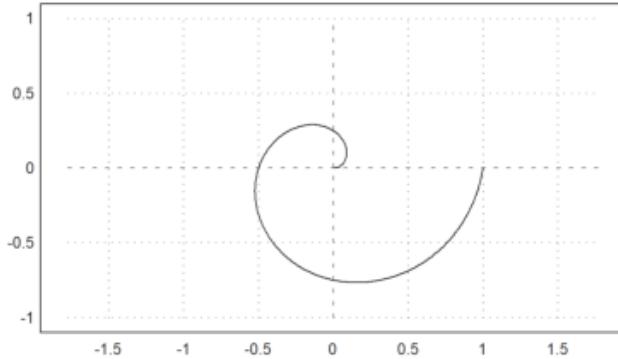
We either need to use very many points for a smooth look or the function adaptive() to evaluate the expressions (see the function adaptive() for more details).

```
>t=linspace(0,1,1000); ...
>plot2d(t*cos(2*pi*t),t*sin(2*pi*t),r=1):
```

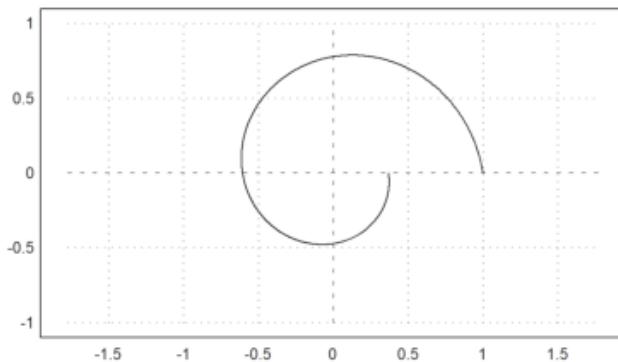


Alternatively, it is possible to use two expressions for curves. The following plots the same curve as above.

```
>plot2d("x*cos(2*pi*x)", "x*sin(2*pi*x)", xmin=0, xmax=1, r=1):
```



```
>t=linspace(0,1,1000); r=exp(-t); x=r*cos(2pi*t); y=r*sin(2pi*t);
>plot2d(x,y,r=1):
```



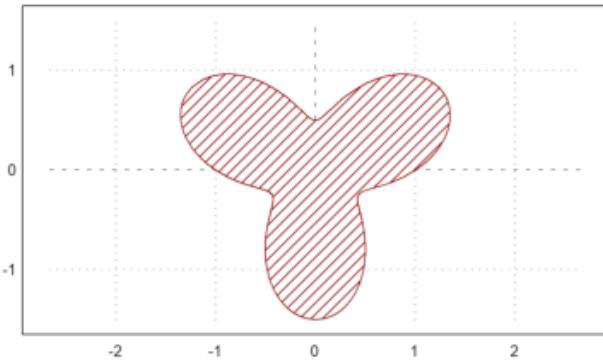
In the next example, we plot the curve

$$\gamma(t) = (r(t) \cos(t), r(t) \sin(t))$$

with

$$r(t) = 1 + \frac{\sin(3t)}{2}.$$

```
>t=linspace(0,2pi,1000); r=1+sin(3*t)/2; x=r*cos(t); y=r*sin(t); ...
>plot2d(x,y,>filled,fillcolor=red,style="/"',r=1.5):
```



Menggambar Grafik Bilangan Kompleks

An array of complex numbers can also be plotted. Then the grid points will be connected. If a number of grid lines is specified (or a 1×2 vector of grid lines) in the argument cgrid only those grid lines are visible.

A matrix of complex numbers will automatically plot as a grid in the complex plane.

In the following example, we plot the image of the unit circle under the exponential function. The cgrid parameter hides some of the grid curves.

```
>aspect(); r=linspace(0,1,50); a=linspace(0,2pi,80)'; z=r*exp(I*a);...
>plot2d(z,a=-1.25,b=1.25,c=-1.25,d=1.25,cgrid=10):
>aspect(1.25); r=linspace(0,1,50); a=linspace(0,2pi,200)'; z=r*exp(I*a);
>plot2d(exp(z),cgrid=[40,10]):
>r=linspace(0,1,10); a=linspace(0,2pi,40)'; z=r*exp(I*a);
>plot2d(exp(z),>points,>add):
```

A vector of complex numbers is automatically plotted as a curve in the complex plane with real part and imaginary part.

In the example, we plot the unit circle with

$$\gamma(t) = e^{it}$$

```
>t=linspace(0,2pi,1000); ...
>plot2d(exp(I*t)+exp(4*I*t),r=2):
```

Statistical Plots

There are many functions which are specialized on statistical plots. One of the often used plots is a column plot.

A cumulative sum of a 0-1-normal distributed values produces a random walk.

```
>plot2d(cumsum(randnormal(1,1000))):
```

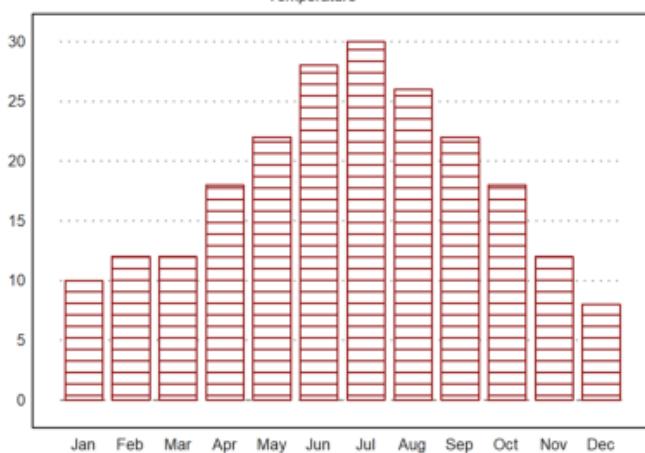


Using two rows shows a walk in two dimensions.

```
>X=cumsum(randnormal(2,1000)); plot2d(X[1],X[2]):  
>columnspplot(cumsum(randnormal(10)),style="/",color=blue):
```

It can also show strings as labels.

```
>months=["Jan", "Feb", "Mar", "Apr", "May", "Jun", ...  
> "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"];  
>values=[10,12,12,18,22,28,30,26,22,18,12,8];  
>columnspplot(values,lab=months,color=red,style="-");  
>title("Temperature"):
```



```
>k=0:10;
>plot2d(k,bin(10,k),>bar):
>plot2d(k,bin(10,k)); plot2d(k,bin(10,k),>points,>add):
>plot2d(normal(1000),normal(1000),>points,grid=6,style=".."):
>plot2d(normal(1,1000),>distribution,style="O"):
>plot2d("qnormal",0,5;2.5,0.5,>filled):
```

To plot an experimental statistical distribution, you can use `distribution=n` with `plot2d`.

```
>w=randexponential(1,1000); // exponential distribution
>plot2d(w,>distribution): // or distribution=n with n intervals
```

Or you can compute the distribution from the data and plot the result with `>bar` in `plot3d`, or with a column plot.

```
>w=normal(1000); // 0-1-normal distribution
>{x,y}=histo(w,10,v=[-6,-4,-2,-1,0,1,2,4,6]); // interval bounds v
>plot2d(x,y,>bar):
```

The `statplot()` function sets the style with a simple string.

```
>statplot(1:10,cumsum(random(10)), "b"):
>n=10; i=0:n; ...
>plot2d(i,bin(n,i)/2^n,a=0,b=10,c=0,d=0.3); ...
>plot2d(i,bin(n,i)/2^n,points=true,style="ow",add=true,color=blue):
```

Moreover, data can be plotted as bars. In this case, x should be sorted and one element longer than y . The bars will extend from $x[i]$ to $x[i+1]$ with values $y[i]$. If x has the same size as y , it will be extended by one element with the last spacing.

Fill styles can be used just as above.

```
>n=10; k=bin(n,0:n); ...
>plot2d(-0.5:n+0.5,k,bar=true,fillcolor=lightgray):
```

The data for bar plots (bar=1) and histograms (histogram=1) can either be explicitly given in xv and yv , or can be computed from an empirical distribution in xv with $>\text{distribution}$ (or $\text{distribution}=n$). Histograms of xv values will be computed automatically with $>\text{histogram}$. If $>\text{even}$ is specified, the xv values will be counted in integer intervals.

```
>plot2d(normal(10000),distribution=50):
>k=0:10; m=bin(10,k); x=(0:11)-0.5; plot2d(x,m,>bar):
>columnsplot(m,k):
>plot2d(random(600)*6,histogram=6):
```

For distributions, there is the parameter $\text{distribution}=n$, which counts values automatically and prints the relative distribution with n sub-intervals.

```
>plot2d(normal(1,1000),distribution=10,style="\\"":
```

With the parameter $\text{even}=true$, this will use integer intervals.

```
>plot2d(intrandom(1,1000,10),distribution=10,even=true):
```

Note that there are many statistical plots, which might be useful. Have a look at the tutorial about statistics.

```
>columnsplot(getmultiplicities(1:6,intrandom(1,6000,6))):
>plot2d(normal(1,1000),>distribution); ...
> plot2d("qnormal(x)",color=red,thickness=2,>add):
```

There are also many special plots for statistics. A boxplot shows the quartiles of this distribution and lots of outliers. By definition, outliers in a boxplot are data which exceed 1.5 times the middle 50% range of the plot.

```
>M=normal(5,1000); boxplot(quartiles(M)):
```

Implicit Functions

Implicit plots show level lines solving $f(x,y)=\text{level}$, where "level" can be a single value or a vector of values. If level="auto", there will be nc level lines, which will spread between the minimum and the maximum of the function evenly. Darker or lighter color can be added with >hue to indicate value of the function. For implicit functions, xv must be a function or an expression of the parameters x and y, or, alternatively, xv can be a matrix of values.

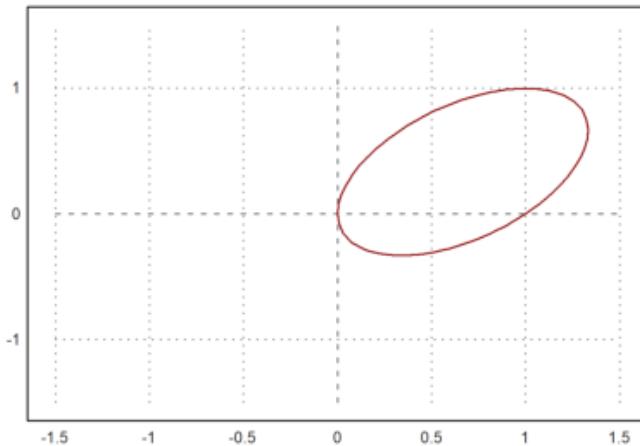
Euler can mark the level lines

$$f(x, y) = c$$

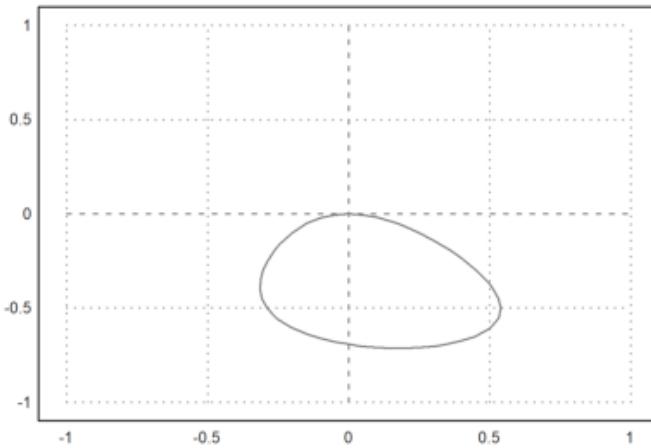
of any function.

To draw the set $f(x,y)=c$ for one or more constants c you can use plot2d() with its implicit plots in the plane. The parameter for c is level=c, where c can be vector of level lines. Additionally, a color scheme can be drawn in the background to indicate the value of the function for each point in the plot. The parameter "n" determines the fineness of the plot.

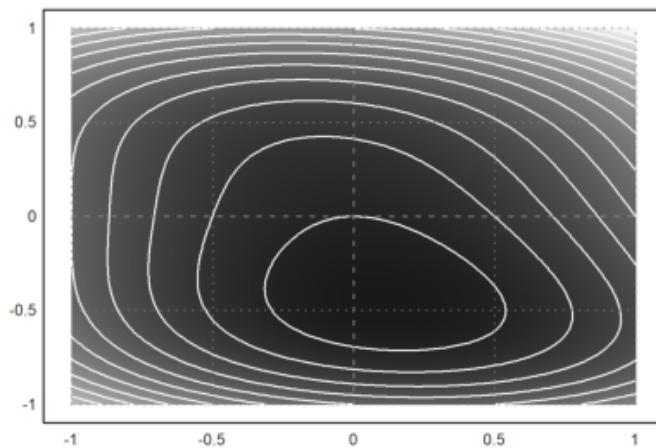
```
>aspect(1.5);
>plot2d("x^2+y^2-x*y-x", r=1.5, level=0, contourcolor=red):
```



```
>expr := "2*x^2+x*y+3*y^4+y"; // define an expression f(x,y)
>plot2d(expr, level=0): // Solutions of f(x,y)=0
```



```
>plot2d(expr,level=0:0.5:20,>hue,contourcolor=white,n=200): // nice
```



```
>plot2d(expr,level=0:0.5:20,>hue,>spectral,n=200,grid=4): // nicer
```

This works for data plots too. But you will have to specify the ranges for the axis labels.

```
>x=-2:0.05:1; y=x'; z=expr(x,y);
>plot2d(z,level=0,a=-1,b=2,c=-2,d=1,>hue):
>plot2d("x^3-y^2",>contour,>hue,>spectral):
>plot2d("x^3-y^2",level=0,contourwidth=3,>add,contourcolor=red):
>z=z+normal(size(z))*0.2;
>plot2d(z,level=0.5,a=-1,b=2,c=-2,d=1):
```

```
>plot2d(expr,level=[0:0.2:5;0.05:0.2:5.05],color=lightgray) :
>plot2d("x^2+y^3+x*y",level=1,r=4,n=100) :
>plot2d("x^2+2*y^2-x*y",level=0:0.1:10,n=100,contourcolor=white,>hue) :
```

It is also possible to fill the set

$$a \leq f(x, y) \leq b$$

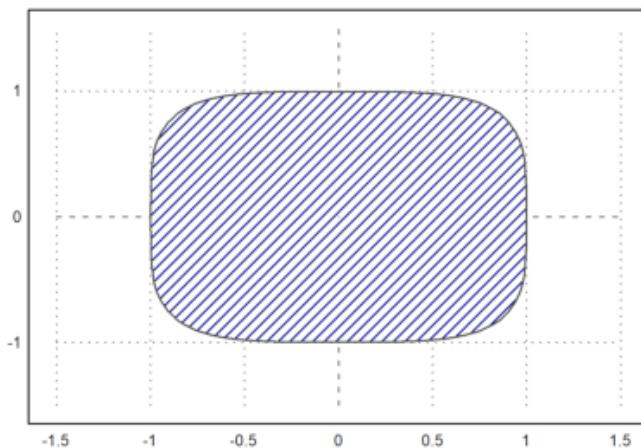
with a level range.

It is possible to fill regions of values for a specific function. For this, level must be a 2xn matrix. The first row are the lower bounds and the second row contains the upper bounds.

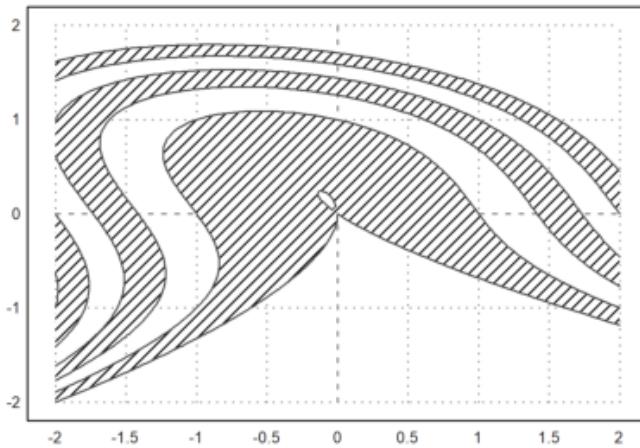
```
>plot2d(expr,level=[0;1],style="-",color=blue) : // 0 <= f(x,y) <= 1
```

Implicit plots can also show ranges of levels. Then level must be a 2xn matrix of level intervals, where the first row contains the start and the second row the end of each interval. Alternatively, a simple row vector can be used for level, and a parameter dl extends the level values to intervals.

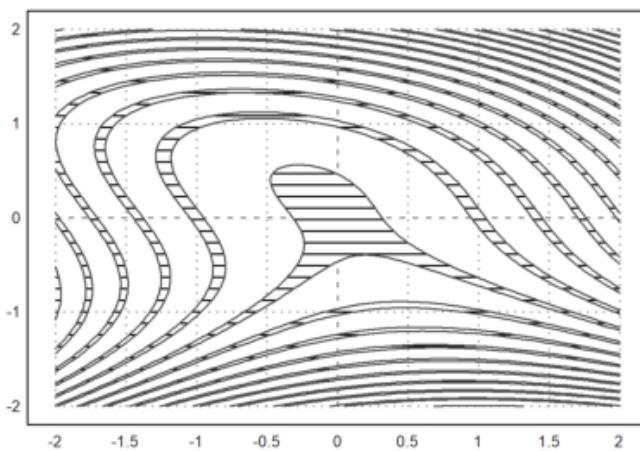
```
>plot2d("x^4+y^4",r=1.5,level=[0;1],color=blue,style="/") :
```



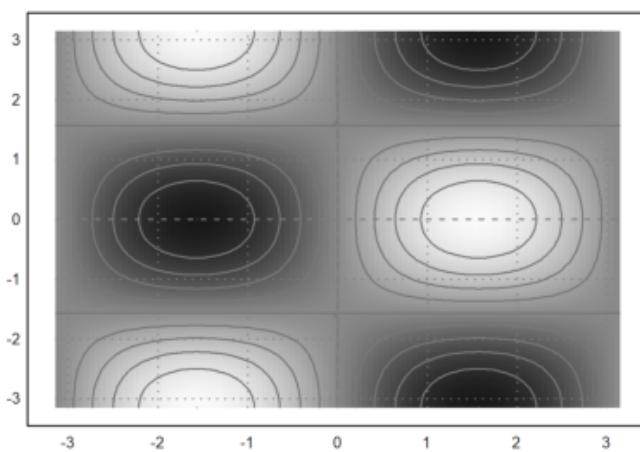
```
>plot2d("x^2+y^3+x*y",level=[0,2,4;1,3,5],style="/",r=2,n=100) :
```



```
>plot2d("x^2+y^3+x*y",level=-10:20,r=2,style="-",dl=0.1,n=100):
```



```
>plot2d("sin(x)*cos(y)",r=pi,>hue,>levels,n=100):
```



It is also possible to mark a region

$$a \leq f(x, y) \leq b.$$

This is done by adding a level with two rows.

```
>plot2d("(x^2+y^2-1)^3-x^2*y^3",r=1.3, ...
> style="#",color=red,<outline, ...
> level=[-2;0],n=100):
```

It is possible to specify a specific level. E.g., we can plot the solution of an equation like

$$x^3 - xy + x^2y^2 = 6$$

```
>plot2d("x^3-x*y+x^2*y^2",r=6,level=1,n=100):
>function starplot1 (v, style="/", color=green, lab=none) ...
```

```
if !holding() then clg; endif;
w=window(); window(0,0,1024,1024);
h=holding(1);
r=max(abs(v))*1.2;
setplot(-r,r,-r,r);
n=cols(v); t=linspace(0,2pi,n);
v=v|v[1]; c=v*cos(t); s=v*sin(t);
cl=barcolor(color); st=barstyle(style);
loop 1 to n
  polygon([0,c[#,c[#+1]], [0,s[#,s[#+1]],1];
  if lab!=none then
    rlab=v[#]+r*0.1;
    {col,row}=toscreen(cos(t[#])*rlab,sin(t[#])*rlab);
    ctext(""+lab#[#],col,row-textheight()/2);
  endif;
end;
barcolor(cl); barstyle(st);
holding(h);
window(w);
endfunction
```

There is no grid or axis ticks here. Moreover, we use the full window for the plot.

We call reset before we test this plot to restore the graphics defaults. This is not necessary, if you are sure that your plot works.

```
>reset; starplot1(normal(1,10)+5,color=red,lab=1:10):
```

Sometimes, you may want to plot something that plot2d cannot do, but almost.

In the following function, we do a logarithmic impulse plot. plot2d can do logarithmic plots, but not for impulse bars.

```
>function logimpulseplot1 (x,y) ...
```

```
{x0,y0}=makeimpulse(x,log(y)/log(10));
plot2d(x0,y0,>bar,grid=0);
h=holding(1);
frame();
xgrid(ticks(x));
p=plot();
for i=-10 to 10;
  if i<=p[4] and i>=p[3] then
    ygrid(i,yt="10^"+i);
  endif;
end;
holding(h);
endfunction
```

Let us test it with exponentially distributed values.

```
>aspect(1.5); x=1:10; y=-log(random(size(x)))*200; ...
>logimpulseplot1(x,y):
```

Let us animate a 2D curve using direct plots. The plot(x,y) command simply plots a curve into the plot window. setplot(a,b,c,d) sets this window.

The wait(0) function forces the plot to appear on the graphics windows. Otherwise, the redraw takes place in sparse time intervals.

```
>function animliss (n,m) ...
```

```
t=linspace(0,2pi,500);
f=0;
c=framecolor(0);
l=linewidth(2);
setplot(-1,1,-1,1);
repeat
```

```

clg;
plot(sin(n*t),cos(m*t+f));
wait(0);
if testkey() then break; endif;
f=f+0.02;
end;
framecolor(c);
linewidth(l);
endfunction

```

Press any key to stop this animation.

```
>animliss(2,3); // lihat hasilnya, jika sudah puas, tekan ENTER
```

Logarithmic Plots

EMT uses the "logplot" parameter for logarithmic scales.

Logarithmic plots can be plotted either using a logarithmic scale in y with logplot=1, or using logarithmic scales in x and y with logplot=2, or in x with logplot=3.

- logplot=1: y-logarithmic
- logplot=2: x-y-logarithmic
- logplot=3: x-logarithmic

```

>plot2d("exp(x^3-x)*x^2",1,5,logplot=1):
>plot2d("exp(x+sin(x))",0,100,logplot=1):
>plot2d("exp(x+sin(x))",10,100,logplot=2):
>plot2d("gamma(x)",1,10,logplot=1):
>plot2d("log(x*(2+sin(x/100)))",10,1000,logplot=3):

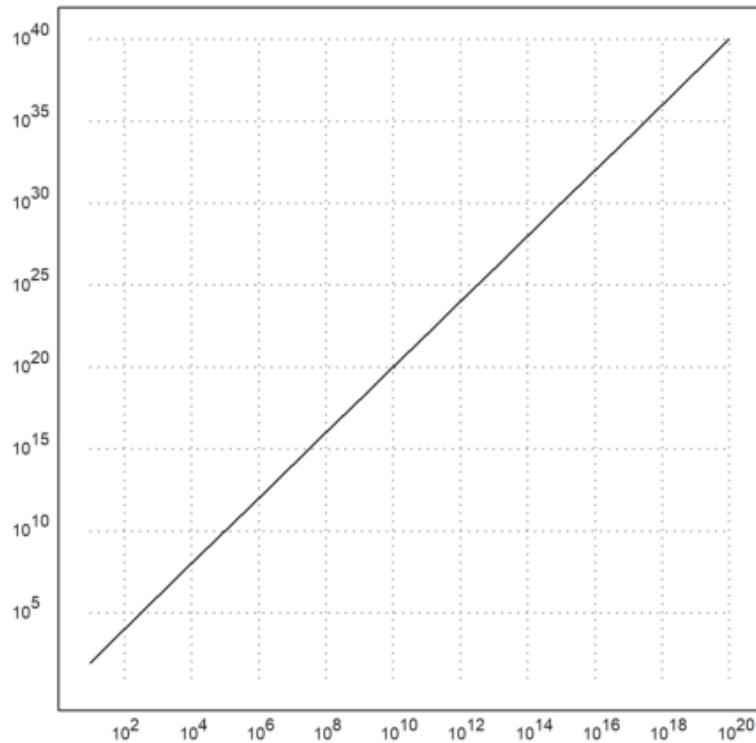
```

This does also work with data plots.

```

>x=10^(1:20); y=x^2-x;
>plot2d(x,y,logplot=2):

```



Contoh Soal

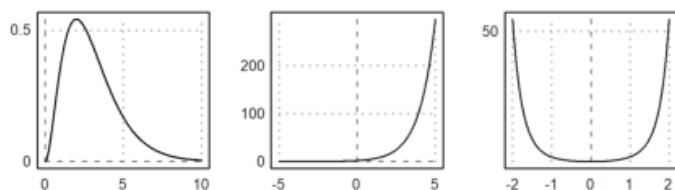
1. Gambarkan plot fungsi-fungsi berikut:

$$f(x) = x^2 e^{-x}, 0 \leq x \leq 10$$

$$g(x) = 2e^x, -5 \leq x \leq 5$$

$$h(x) = e^{x^2}, -2 \leq x \leq 2$$

```
>reset;
>aspect(3,1);
>figure(1,3);...
>figure(1); plot2d("x^2*exp(-x)",0,10);...
>figure(2); plot2d("2*exp(x)",-5,5);...
>figure(3); plot2d("exp(x^2)",-2,2);...
>figure(0):
```

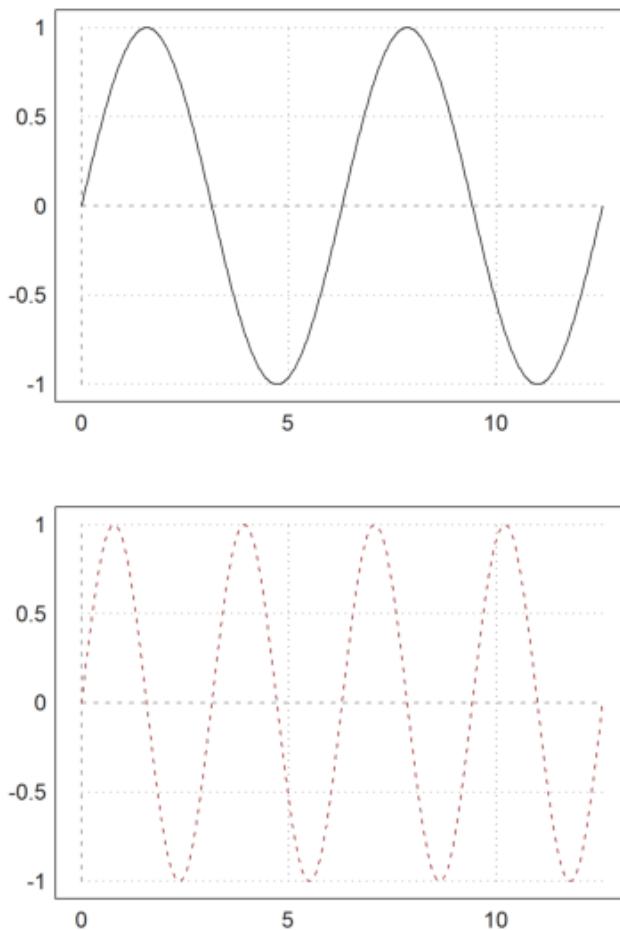


2. Gambarkan kedua fungsi di bawah ini dengan grid yang berbeda

$$f(x) = \sin(x), 0 \leq x \leq 4\pi$$

$$g(x) = \sin(2x), 0 \leq x \leq 4\pi$$

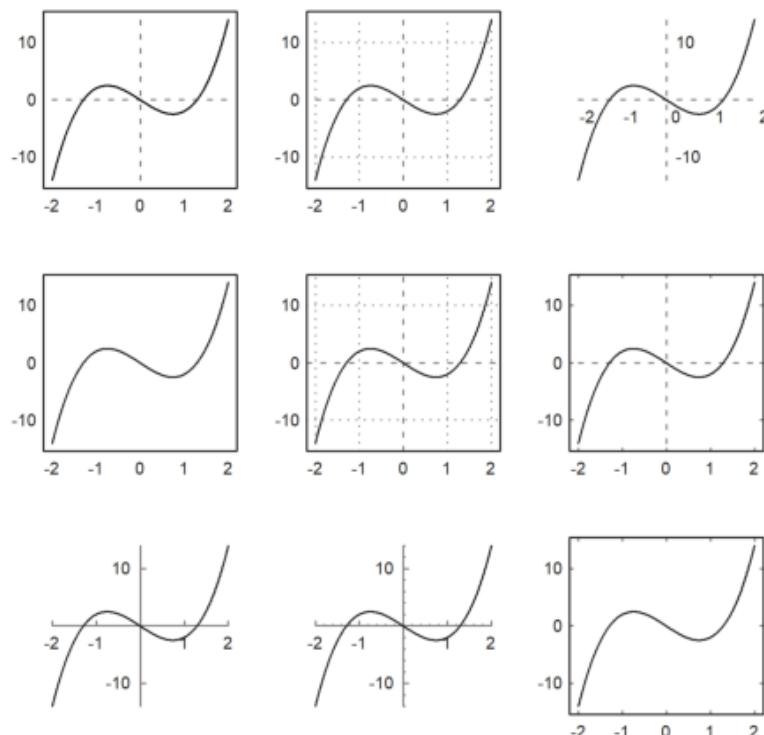
```
>reset;
>aspect(2,3);
>figure(2,1);...
>figure(1); plot2d("sin(x)",0,4pi,grid=2);...
>figure(2); plot2d("sin(2x)",0,4pi, grid=2, color=red, style="--", );
>figure(0):
```



3. Gambarkan plot fungsi di bawah ini dengan 9 jenis grid yang berbeda

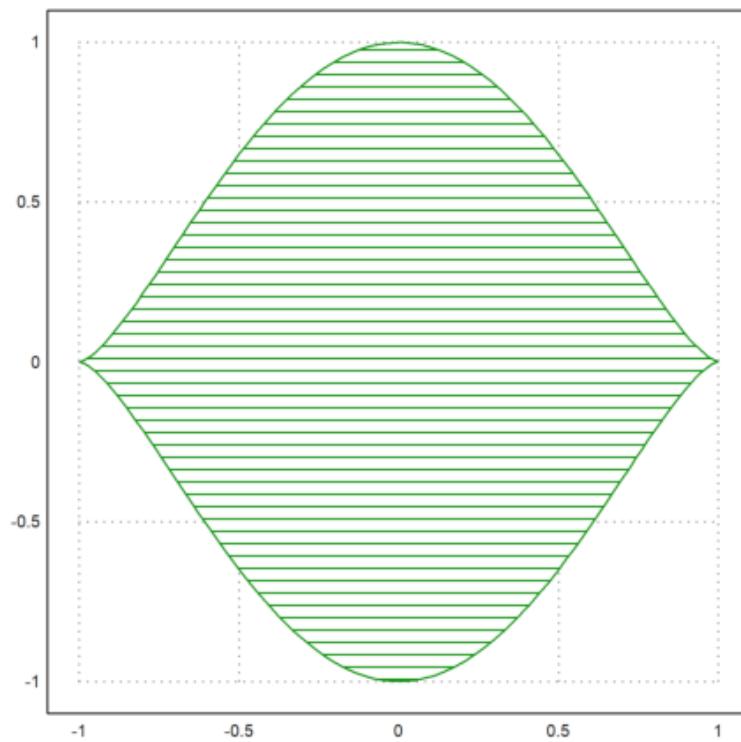
$$f(x) : 3x^3 - 5x; -2 \leq x \leq 2$$

```
>reset;
>figure(3,3);...
>for k=1:9; figure(k); plot2d("3x^3-5x",-2,2,grid=k); end;...
>figure(0):
```



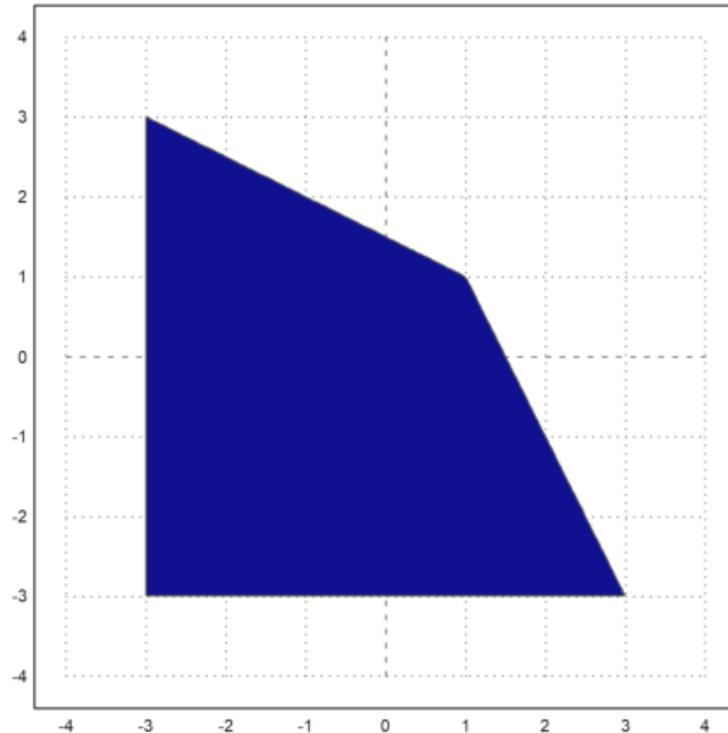
4. Gambarkan plot

```
>reset;
>plot2d("cos(x)","sin(x)^3",xmin=0,xmax=2pi,>filled,style="-"):
```



5. Gambarkan plot fungsi

```
>A=[2,1;1,2;-1,0;0,-1];
>function f(x,y) := max([x,y].A');
>plot2d("f",r=4,level=[0;3],color=blue,n=111):
```



Rujukan Lengkap Fungsi plot2d()

```
function plot2d (xv, yv, btest, a, b, c, d, xmin, xmax, r, n, ..
logplot, grid, frame, framecolor, square, color, thickness, style, ..
auto, add, user, delta, points, addpoints, pointstyle, bar, histogram,
distribution, even, steps, own, adaptive, hue, level, contour, ..
nc, filled, fillcolor, outline, title, xl, yl, maps, contourcolor, ..
contourwidth, ticks, margin, clipping, cx, cy, insimg, spectral, ..
cgrid, vertical, smaller, dl, niveau, levels)
```

Multipurpose plot function for plots in the plane (2D plots). This function can do plots of functions of one variables, data plots, curves in the plane, bar plots, grids of complex numbers, and implicit plots of functions of two variables.

Parameters

x,y : equations, functions or data vectors

a,b,c,d : Plot area (default a=-2,b=2)

r : if r is set, then a=cx-r, b=cx+r, c=cy-r, d=cy+r

xmin,xmax : range of the parameter for curves

auto : Determine y-range automatically (default)

square : if true, try to keep square x-y-ranges

n : number of intervals (default is adaptive)

grid : 0 = no grid and labels,

```

1 = axis only,
2 = normal grid (see below for the number of grid lines)
3 = inside axis
4 = no grid
5 = full grid including margin
6 = ticks at the frame
7 = axis only
8 = axis only, sub-ticks

```

frame : 0 = no frame

framecolor: color of the frame and the grid

margin : number between 0 and 0.4 for the margin around the plot

color : Color of curves. If this is a vector of colors,

it will be used for each row of a matrix of plots. In the case point plots, it should be a column vector. If a row vector or a full matrix of colors is used for point plots, it will be used each data point.

thickness : line thickness for curves

This value can be smaller than 1 for very thin lines.

style : Plot style for lines, markers, and fills.

For points use

```

"[ ]", "<>", ". .", ". . .", ". . . .",
"*", "+", "|", "-", "o"
"[ ]#", "<>#", "o#" (filled shapes)
"[ ]w", "<>w", "ow" (non-transparent)

```

For lines use

```

"-", "--", "-.", ". .", ".-. .", "-.-", "->"

```

For filled polygons or bar plots use

```

"#", "#o", "o", "/", "\\", "\\", 
"+", "|", "-", "t"

```

points : plot single points instead of line segments
 addpoints : if true, plots line segments and points
 add : add the plot to the existing plot
 user : enable user interaction for functions
 delta : step size for user interaction
 bar : bar plot (x are the interval bounds, y the interval values)
 histogram : plots the frequencies of x in n subintervals
 distribution=n : plots the distribution of x with n subintervals
 even : use inter values for automatic histograms.
 steps : plots the function as a step function (steps=1,2)
 adaptive : use adaptive plots (n is the minimal number of steps)
 level : plot level lines of an implicit function of two variables
 outline : draws boundary of level ranges.
 If the level value is a 2xn matrix, ranges of levels will be drawn
 in the color using the given fill style. If outline is true, it
 will be drawn in the contour color. Using this feature, regions of
 $f(x,y)$ between limits can be marked.
 hue : add hue color to the level plot to indicate the function

value

contour : Use level plot with automatic levels
 nc : number of automatic level lines
 title : plot title (default "")
 xl, yl : labels for the x- and y-axis
 smaller : if >0, there will be more space to the left for labels.
 vertical :

Turns vertical labels on or off. This changes the global variable
 verticallabels locally for one plot. The value 1 sets only vertical
 text, the value 2 uses vertical numerical labels on the y axis.

filled : fill the plot of a curve
 fillcolor : fill color for bar and filled curves
 outline : boundary for filled polygons
 logplot : set logarithmic plots

```

1 = logplot in y,
2 = logplot in xy,
3 = logplot in x
  
```

own :

A string, which points to an own plot routine. With >user, you get the same user interaction as in plot2d. The range will be set before each call to your function.

maps : map expressions (0 is faster), functions are always mapped.

contourcolor : color of contour lines

contourwidth : width of contour lines

clipping : toggles the clipping (default is true)

title :

This can be used to describe the plot. The title will appear above the plot. Moreover, a label for the x and y axis can be added with xl="string" or yl="string". Other labels can be added with the functions label() or labelbox(). The title can be a unicode string or an image of a Latex formula.

cgrid :

Determines the number of grid lines for plots of complex grids. Should be a divisor of the the matrix size minus 1 (number of subintervals). cgrid can be a vector [cx,cy].

Overview

The function can plot

- expressions, call collections or functions of one variable,
- parametric curves,
- x data against y data,
- implicit functions,
- bar plots,
- complex grids,
- polygons.

If a function or expression for xv is given, plot2d() will compute values in the given range using the function or expression. The expression must be an expression in the variable x. The range must be defined in the parameters a and b unless the default range should be used. The y-range will be computed automatically, unless c and d are specified, or a radius r, which yields the range r,r

for x and y. For plots of functions, plot2d will use an adaptive evaluation of the function by default. To speed up the plot for complicated functions, switch this off with <adaptive, and optionally decrease the number of intervals n. Moreover, plot2d() will by default use mapping. I.e., it will compute the plot element

for element. If your expression or your functions can handle a vector x , you can switch that off with `<maps` for faster evaluation.

Note that adaptive plots are always computed element for element. If functions or expressions for both xv and for yv are specified, `plot2d()` will compute a curve with the xv values as x-coordinates and the yv values as y-coordinates. In this case, a range should be defined for the parameter using `xmin`, `xmax`. Expressions contained in strings must always be expressions in the parameter variable x .

BAB

Menggambar Plot 3D dengan EMT

Ini adalah pengenalan plot 3D di Euler. Kita memerlukan plot 3D untuk memvisualisasikan fungsi dari dua variabel.

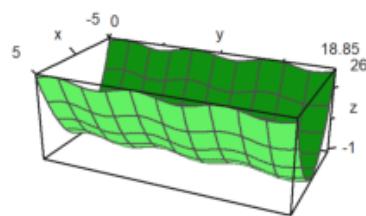
Euler menggambar fungsi-fungsi tersebut dengan menggunakan algoritme pengurutan untuk menyembunyikan bagian-bagian di latar belakang. Secara umum, Euler menggunakan proyeksi pusat. Standarnya adalah dari kuadran x-y positif ke arah asal $x=y=z=0$, tetapi sudut=0° terlihat dari arah sumbu-y. Sudut pandang dan ketinggian dapat diubah.

Euler dapat mem-plot :

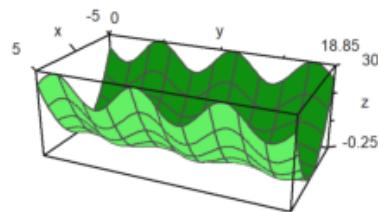
- memplot - permukaan dengan garis bayangan dan garis datar,
- awan titik,
- kurva parametrik,
- permukaan implisit.

Plot 3D suatu fungsi menggunakan plot3d. Cara termudah adalah dengan memplot ekspresi dalam x dan y. Parameter r mengatur rentang plot sekitar (0,0).

```
>aspect(1.5); plot3d("x^2+sin(y)", -5, 5, 0, 6*pi):
```

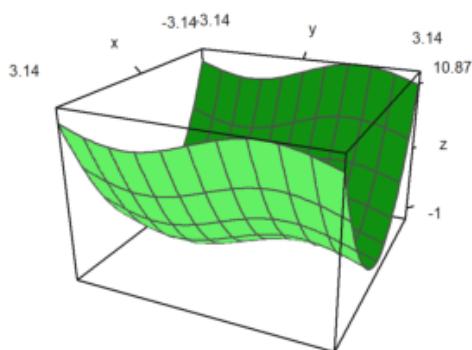


```
>plot3d("x^2+x*sin(y)", -5, 5, 0, 6*pi) :
```



Silakan lakukan modifikasi agar gambar "talang bergelombang" tersebut tidak lurus melainkan melengkung/melingkar, baik melingkar secara mendatar maupun melingkar turun/naik (seperti papan peluncur pada kolam renang). Temukan rumusnya.

```
>aspect(1.5); plot3d("x^2+sin(y)", r=pi) :
```



Untuk grafik suatu fungsi, gunakan

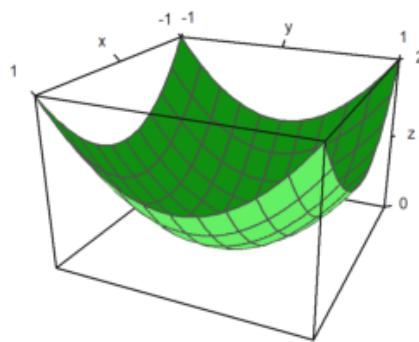
- ekspresi sederhana dalam x dan y,
- nama fungsi dari dua variabel,
- atau matriks data.

Standarnya adalah kisi-kisi kawat berisi dengan warna berbeda di kedua sisi. Perhatikan bahwa jumlah interval kisi default adalah 10, tetapi plot menggunakan jumlah default persegi panjang 40x40 untuk membuat permukaannya. Ini bisa diubah.

- n=40, n=[40,40]: jumlah garis grid di setiap arah.
- grid=10, grid=[10,10]: : jumlah garis grid di setiap arah.

Kita menggunakan default n=40 dan grid=10.

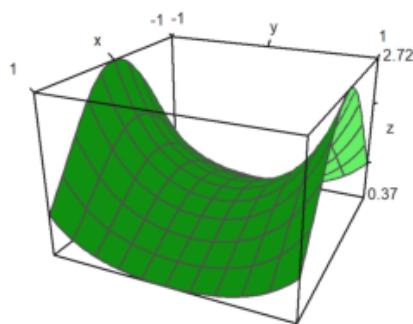
```
>plot3d("x^2+y^2") :
```



Interaksi pengguna dimungkinkan dengan parameter >pengguna. Pengguna dapat menekan tombol berikut.

- left,right,up,down: : putar sudut pandang,
- +,-: zoom in or out
- a: menghasilkan anaglyph (lihat di bawah)
- l: beralih memutar sumber cahaya(lihat di bawah)
- space: reset to default
- return: end interaction

```
>plot3d("exp(-x^2+y^2)",>user, ...
> title="Turn with the vector keys (press return to finish)":
```



Rentang plot untuk fungsi dapat ditentukan dengan :

- a,b: the x-range
- c,d: the y-range
- r: a symmetric square around (0,0).
- n: number of subintervals for the plot.

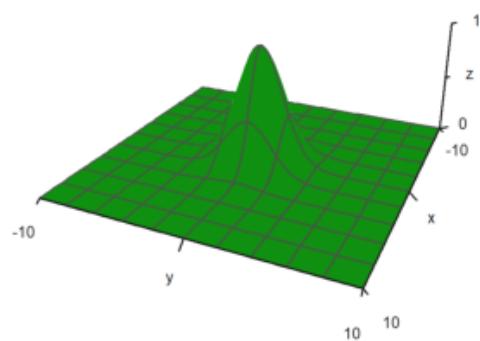
Ada beberapa parameter untuk menskalakan fungsi atau mengubah tampilan grafik.

fscale: scales to function values (default is <fscale>).

scale: angka atau vektor 1x2 untuk menskalakan ke arah x dan y

frame: jenis bingkai (default 1).

```
>plot3d("exp(-(x^2+y^2)/5)",r=10,n=80,fscale=4,scale=1.2,frame=3,>user):
```



Tampilan dapat diubah dengan berbagai cara.

- distance: jarak pandang ke plot.
- zoom: the zoom value.
- sudut: the angle to the negative y-axis in radians.
- height: the height of the view in radians.

Nilai default dapat diperiksa atau diubah dengan fungsi `view()`. Ini mengembalikan parameter dalam urutan di atas.

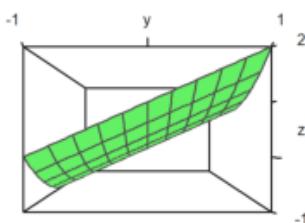
```
>view
```

```
[5, 2.6, 2, 0.4]
```

Jarak yang lebih dekat membutuhkan lebih sedikit zoom. Efeknya lebih seperti lensa sudut lebar.

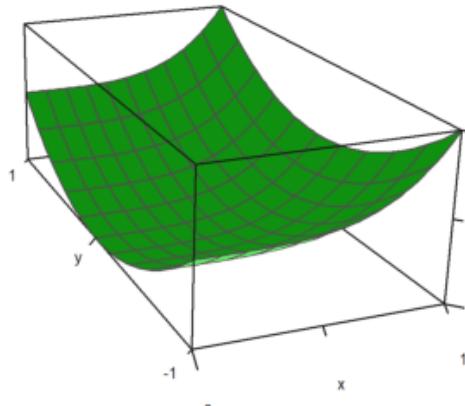
ada contoh berikut, sudut=0 dan tinggi=0 dilihat dari sumbu y negatif. Label sumbu untuk y disembunyikan dalam kasus ini.

```
>plot3d("x^2+y",distance=3,zoom=1,angle=pi/2,height=0):
```



Plot selalu terlihat berada di tengah kubus plot. Anda dapat memindahkan bagian tengah dengan parameter tengah.

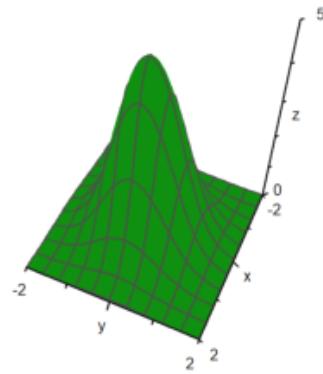
```
>plot3d("x^4+y^2",a=0,b=1,c=-1,d=1,angle=-20°,height=20°, ...
> center=[0.4,0,0],zoom=5):
```



Plotnya diskalakan agar sesuai dengan unit kubus untuk dilihat. Jadi tidak perlu mengubah jarak atau zoom tergantung ukuran plot. Namun labelnya mengacu pada ukuran sebenarnya.

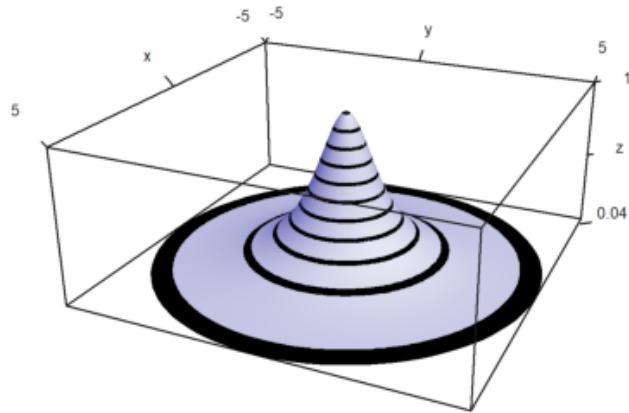
Jika Anda mematikannya dengan `scale=false`, Anda harus berhati-hati agar plot tetap masuk ke dalam jendela plotting, dengan mengubah jarak pandang atau zoom, dan memindahkan bagian tengah.

```
>plot3d("5*exp(-x^2-y^2)", r=2, <fscale, <scale, distance=13, height=50°, ...
> center=[0, 0, -2], frame=3):
```

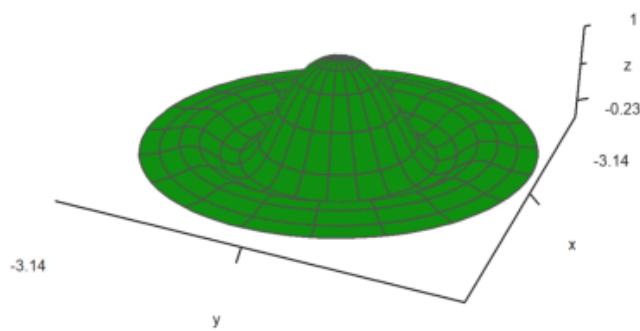


Plot kutub juga tersedia. Parameter `polar=true` menggambar plot kutub. Fungsi tersebut harus tetap merupakan fungsi dari x dan y. Parameter "fscale" menskalakan fungsi dengan skalanya sendiri. Kalau tidak, fungsinya akan diskalakan agar sesuai dengan kubus.

```
>plot3d("1/(x^2+y^2+1)", r=5, >polar, ...
>fscale=2, >hue, n=100, zoom=4, >contour, color=blue):
```



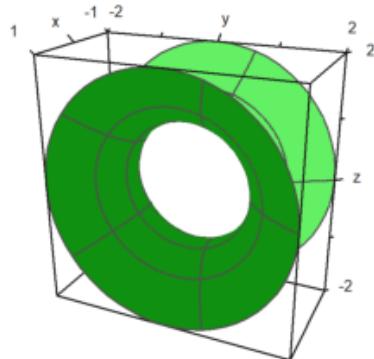
```
>function f(r) := exp(-r/2)*cos(r); ...
>plot3d("f(x^2+y^2)",>polar,scale=[1,1,0.4],r=pi,frame=3,zoom=4):
```



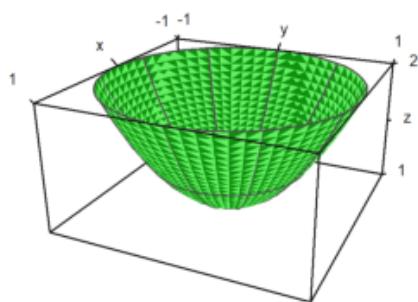
Parameter memutar memutar fungsi di x di sekitar sumbu x.

- rotate=1: Uses the x-axis
- rotate=2: Uses the z-axis

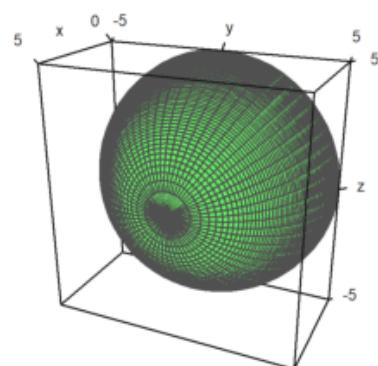
```
>plot3d("x^2+1",a=-1,b=1,rotate=true,grid=5):
```



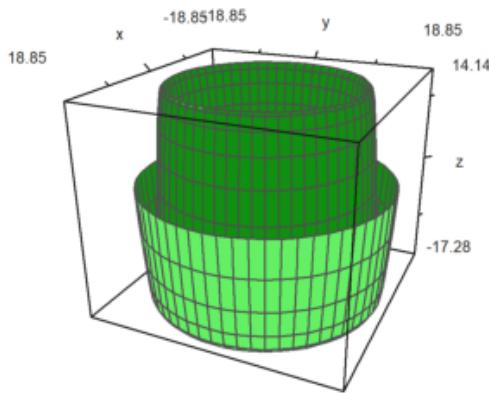
```
>plot3d("x^2+1",a=-1,b=1,rotate=2,grid=5):
```



```
>plot3d("sqrt(25-x^2)",a=0,b=5,rotate=1):
```

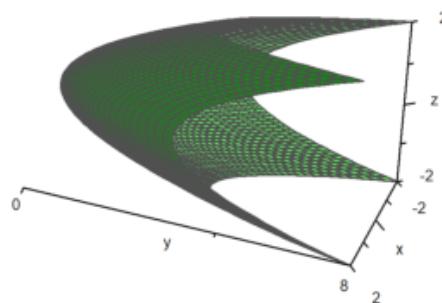


```
>plot3d("x*sin(x)", a=0, b=6pi, rotate=2) :
```



Berikut adalah plot dengan tiga fungsi.

```
>plot3d("x", "x^2+y^2", "y", r=2, zoom=3.5, frame=3) :
```



Plot Kontur

Untuk plotnya, Euler menambahkan garis grid. Sebaliknya dimungkinkan untuk menggunakan garis datar dan rona satu warna atau rona warna spektral. Euler dapat menggambar ketinggian fungsi pada plot dengan arsiran. Di semua plot 3D, Euler dapat menghasilkan anaglyph.

- >hue: Mengaktifkan bayangan cahaya, bukan kabel.

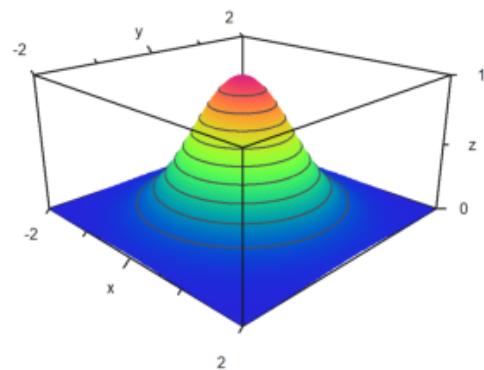
>contour: : Membuat plot garis kontur otomatis pada plot.

- level=... (or levels): A Vektor nilai garis kontur.

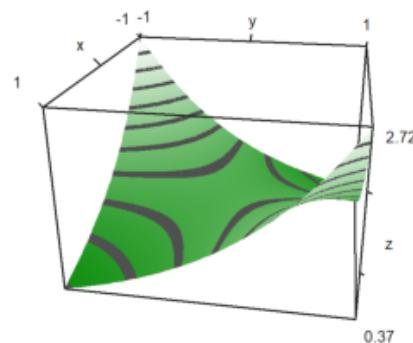
Standarnya adalah level="auto", yang menghitung beberapa garis level secara otomatis. Seperti yang Anda lihat di plot, level sebenarnya adalah rentang level.

Gaya default dapat diubah. Untuk plot kontur berikut, kami menggunakan grid yang lebih halus berukuran 100x100 poin, menskalakan fungsi dan plot, dan menggunakan sudut pandang yang berbeda.

```
>plot3d("exp(-x^2-y^2)",r=2,n=100,level="thin", ...
> >contour,>spectral,fscale=1,scale=1.1,angle=45°,height=20°) :
```



```
>plot3d("exp(x*y)",angle=100°,>contour,color=green) :
```

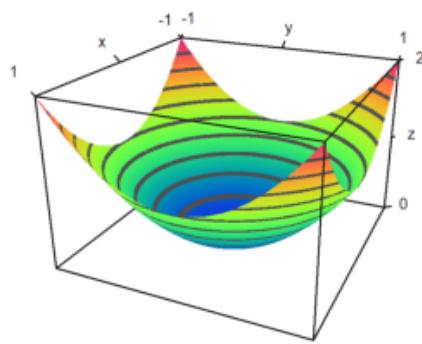


Bayangan defaultnya menggunakan warna abu-abu. Namun rentang warna spektral juga tersedia.

- >spectral: Menggunakan skema spektral default
- color=...: Menggunakan warna khusus atau skema

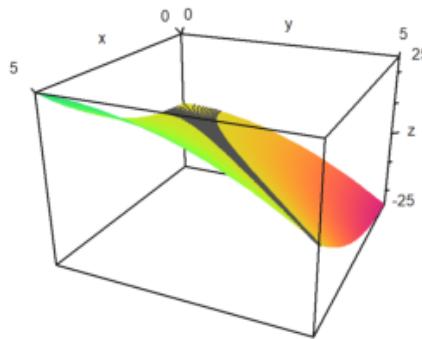
Untuk plot berikut, kami menggunakan skema spektral default dan menambah jumlah titik untuk mendapatkan tampilan yang sangat halus.

```
>plot3d("x^2+y^2",>spectral,>contour,n=100):
```



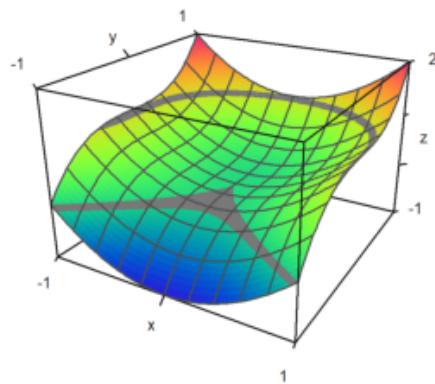
Selain garis level otomatis, kita juga dapat menetapkan nilai garis level. Ini akan menghasilkan garis level yang tipis, bukan rentang level.

```
>plot3d("x^2-y^2",0,5,0,5,level=-1:0.1:1,color=redgreen):
```



Dalam plot berikut, kita menggunakan dua pita tingkat yang sangat luas dari -0,1 hingga 1, dan dari 0,9 hingga 1. Ini dimasukkan sebagai matriks dengan batas tingkat sebagai kolom. Selain itu, kami melapisi grid dengan 10 interval di setiap arah.

```
>plot3d("x^2+y^3",level=[-0.1,0.9;0,1], ...
> >spectral,angle=30°,grid=10,contourcolor=gray):
```

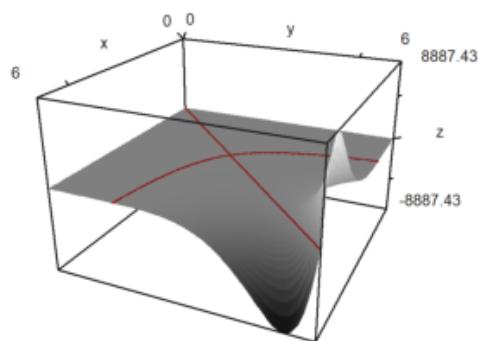


Pada contoh berikut, kita memplot himpunan, di mana :

$$f(x, y) = x^y - y^x = 0$$

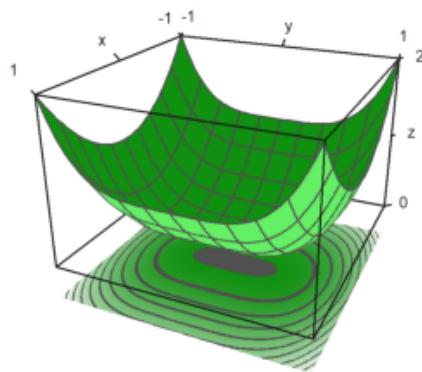
Kami menggunakan satu garis tipis untuk garis level.

```
>plot3d("x^y-y^x",level=0,a=0,b=6,c=0,d=6,contourcolor=red,n=100):
```



Dimungkinkan untuk menampilkan bidang kontur di bawah plot. Warna dan jarak ke plot dapat ditentukan.

```
>plot3d("x^2+y^4", >cp, cpcolor=green, cpdelta=0.2) :
```



Berikut beberapa gaya lainnya. Kami selalu mematikan bingkai, dan menggunakan berbagai skema warna untuk plot dan kisi.

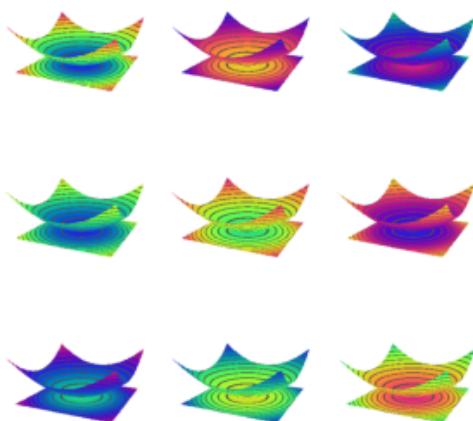
```
>figure(2,2); ...
>expr="y^3-x^2"; ...
>figure(1); ...
> plot3d(expr,<frame,>cp, cpcolor=spectral); ...
>figure(2); ...
> plot3d(expr,<frame,>spectral,grid=10,cp=2); ...
>figure(3); ...
> plot3d(expr,<frame,>contour,color=gray,nc=5, cp=3, cpcolor=greenred); ...
>figure(4); ...
> plot3d(expr,<frame,>hue,grid=10,>transparent,>cp, cpcolor=gray); ...
>figure(0) :
```



Ada beberapa skema spektral lainnya, yang diberi nomor dari 1 hingga 9. Namun Anda juga dapat menggunakan warna=nilai, di mana nilai :

- spectral: untuk rentang dari biru ke merah
- white: untuk rentang yang lebih redup
- yellowblue,purplegreen,blueyellow,greenred
- blueyellow, greenpurple,yellowblue,redgreen

```
>figure(3,3); ...
>for i=1:9; ...
> figure(i); plot3d("x^2+y^2",spectral=i,>contour,>cp,<frame,zoom=4); ...
>end; ...
>figure(0):
```



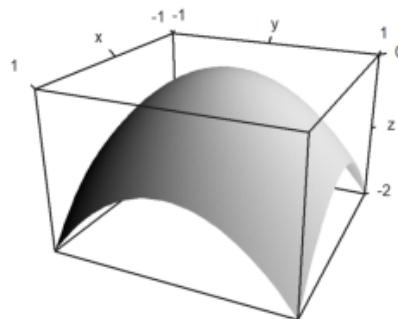
Sumber cahaya dapat diubah dengan 1 dan tombol cursor selama interaksi pengguna. Itu juga dapat diatur dengan parameter.

- light: arah
- amb: ambient light between 0 and 1

Catatan : program tidak membuat perbedaan antara sisi plot. Tidak ada bayangan. Untuk ini, Anda memerlukan Povray.

```
>plot3d("-x^2-y^2", ...
>  hue=true, light=[0,1,1], amb=0, user=true, ...
>  title="Press l and cursor keys (return to exit)":
```

Press l and cursor keys (return to exit)



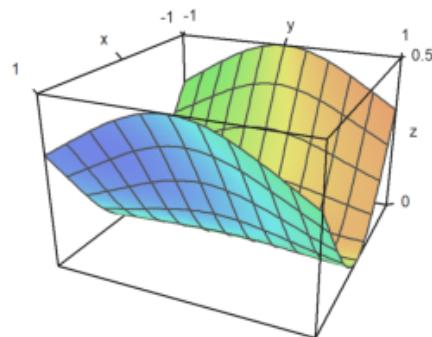
Parameter warna mengubah warna permukaan. Warna garis level juga bisa diubah.

```
>plot3d("-x^2-y^2",color=rgb(0.2,0.2,0),hue=true,frame=false, ...
>  zoom=3,contourcolor=red,level=-2:0.1:1,dl=0.01):
```



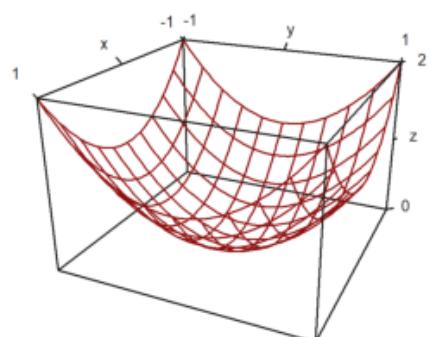
The color 0 gives a special rainbow effect.

```
>plot3d("x^2/(x^2+y^2+1)",color=0,hue=true,grid=10):
```



The surface can also be transparent.

```
>plot3d("x^2+y^2",>transparent,grid=10,wirecolor=red):
```



Ada juga plot implisit dalam tiga dimensi. Euler menghasilkan pemotongan melalui objek. Fitur plot3d mencakup plot implisit. Plot ini menunjukkan himpunan nol suatu fungsi dalam tiga variabel. Permukaannya juga bisa transparan.

Solusi dari

Latex: $f(x,y,z) = 0$

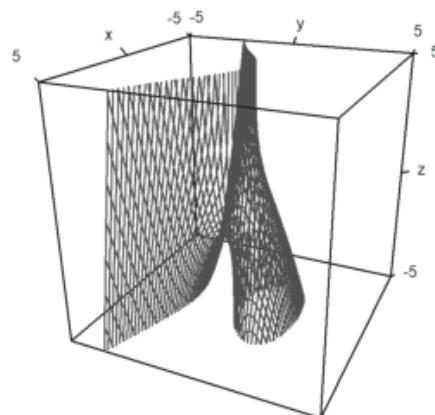
dapat divisualisasikan dalam potongan yang sejajar dengan bidang xy-, xz- dan yz.

- implicit=1: cut parallel to the y-z-plane
- implicit=2: cut parallel to the x-z-plane
- implicit=4: cut parallel to the x-y-plane

Tambahkan nilai berikut, jika Anda mau. Dalam contoh kita memplot :

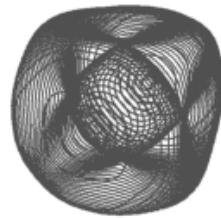
$$M = \{(x, y, z) : x^2 + y^3 + zy = 1\}$$

```
>plot3d("x^2+y^3+z*y-1", r=5, implicit=3):
```



```
>c=1; d=1;
```

```
>plot3d("((x^2+y^2-c^2)^2+(z^2-1)^2)*((y^2+z^2-c^2)^2+(x^2-1)^2)*((z^2+x^2-
```



Cannot combine a 41x41 and a 1x81 matrix for +!

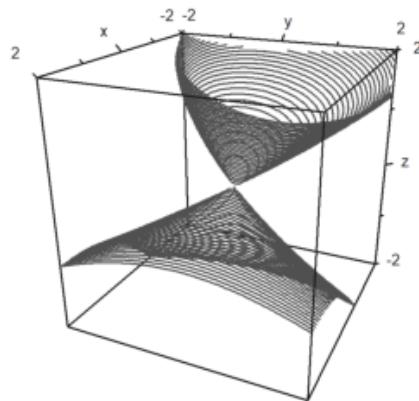
Error in expression: $((x^2+y^2-c^2)^2+(z^2-1)^2)*((y^2+z^2-c^2)^2+(x^2-1)^2)$

Try "trace errors" to inspect local variables after errors.

pov3d:

```
z=f(x,y;args());
```

```
>plot3d("x^2+y^2+4*x*z+z^3",>implicit,r=2,zoom=2.5):
```



Merencanakan Data 3D

Sama seperti plot2d, plot3d menerima data. Untuk objek 3D, Anda perlu menyediakan matriks nilai x-, y- dan z, atau tiga fungsi atau ekspresi $fx(x,y)$, $fy(x,y)$, $fz(x,y)$.

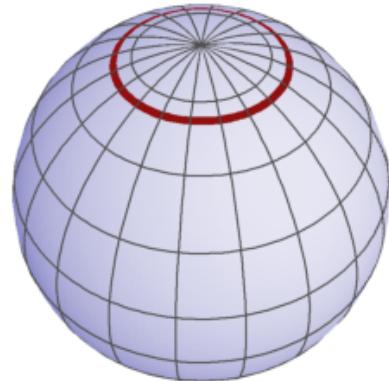
$$\gamma(t, s) = (x(t, s), y(t, s), z(t, s))$$

Karena x, y, z adalah matriks, kita asumsikan bahwa (t, s) melewati grid persegi. Hasilnya, Anda dapat memplot gambar persegi panjang di ruang angkasa.

Anda dapat menggunakan bahasa matriks Euler untuk menghasilkan koordinat secara efektif.

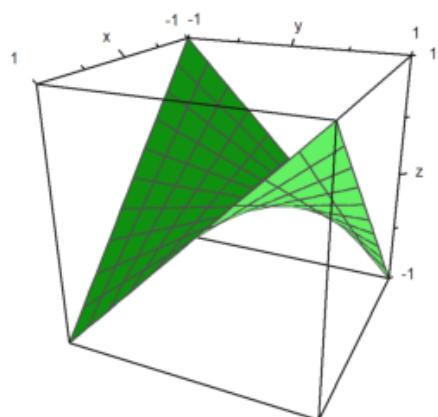
Dalam contoh berikut, kita menggunakan vektor nilai t dan vektor kolom nilai s untuk membuat parameter permukaan bola. Dalam gambar kita dapat menandai wilayah, dalam kasus kita wilayah kutub.

```
>t=linspace(0,2pi,180); s=linspace(-pi/2,pi/2,90)'; ...
>x=cos(s)*cos(t); y=cos(s)*sin(t); z=sin(s); ...
>plot3d(x,y,z,>hue, ...
>color=blue,<frame,grid=[10,20], ...
>values=s,contourcolor=red,level=[90°-24°;90°-22°], ...
>scale=1.4,height=50°):
```



Here is an example, which is the graph of a function.

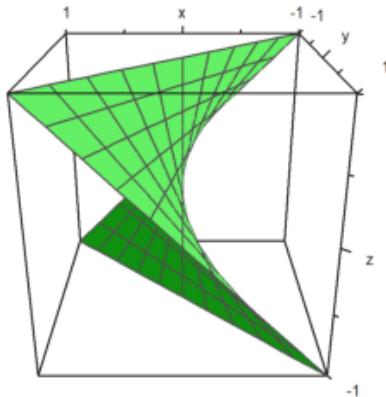
```
>t=-1:0.1:1; s=(-1:0.1:1)'; plot3d(t,s,t*s,grid=10):
```



Namun, kita bisa membuat berbagai macam permukaan. Berikut adalah permukaan yang sama sebagai suatu fungsi :

$$x = y z$$

```
>plot3d(t*s,t,s,angle=180°,grid=10) :
```



Dengan lebih banyak usaha, kita dapat menghasilkan banyak permukaan.

alam contoh berikut kita membuat tampilan bayangan dari bola yang terdistorsi. Koordinat bola yang biasa adalah

$$\gamma(t, s) = (\cos(t) \cos(s), \sin(t) \sin(s), \cos(s))$$

with

$$0 \leq t \leq 2\pi, \quad -\frac{\pi}{2} \leq s \leq \frac{\pi}{2}.$$

Kami mendistorsi ini dengan sebuah faktor

$$d(t, s) = \frac{\cos(4t) + \cos(8s)}{4}.$$

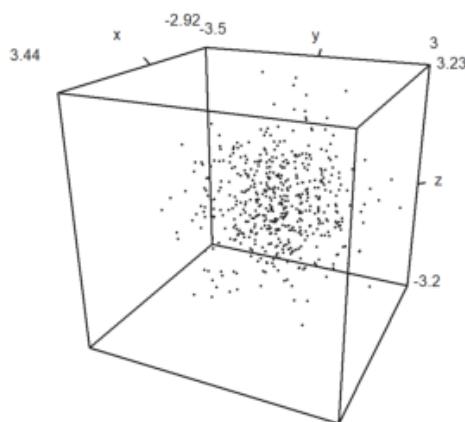
```
>t=linspace(0,2pi,320); s=linspace(-pi/2,pi/2,160)'; ...
>d=1+0.2*(cos(4*t)+cos(8*s)); ...
>plot3d(cos(t)*cos(s)*d,sin(t)*cos(s)*d,sin(s)*d,hue=1, ...
> light=[1,0,1],frame=0,zoom=5):
```



Tentu saja, point cloud juga dimungkinkan. Untuk memplot data titik dalam ruang, kita memerlukan tiga vektor untuk koordinat titik-titik tersebut.

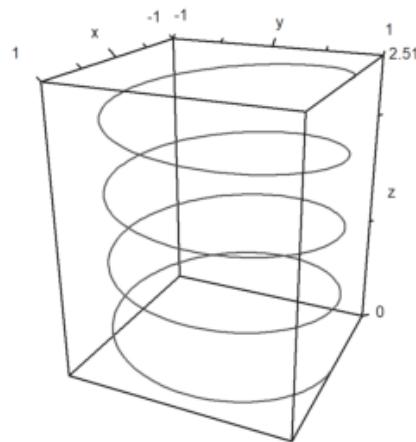
Gayanya sama seperti di plot2d dengan points=true;

```
>n=500; ...
> plot3d(normal(1,n),normal(1,n),normal(1,n),points=true,style=".") :
```

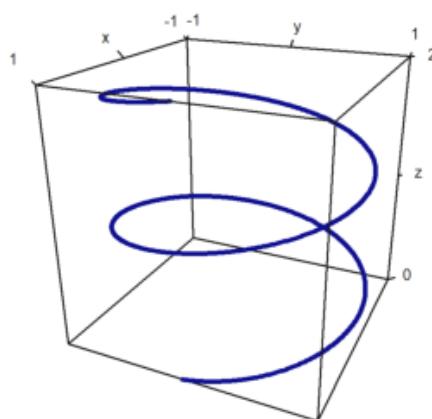


Dimungkinkan juga untuk memplot kurva dalam 3D. Dalam hal ini, lebih mudah untuk menghitung terlebih dahulu titik-titik kurva. Untuk kurva pada bidang kita menggunakan barisan koordinat dan parameter wire=true.

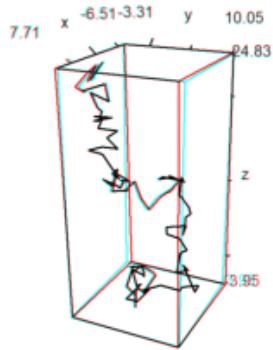
```
>t=linspace(0,8pi,500); ...
>plot3d(sin(t),cos(t),t/10,>wire,zoom=3) :
```



```
>t=linspace(0,4pi,1000); plot3d(cos(t),sin(t),t/2pi,>wire, ...
>lineWidth=3, wirecolor=blue):
```

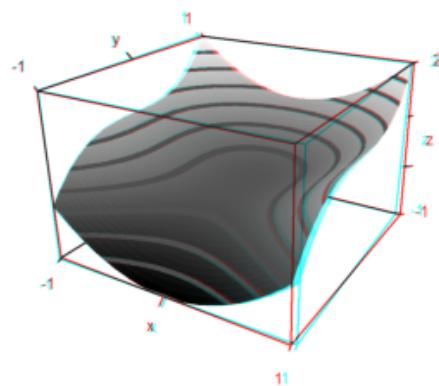


```
>X=cumsum(normal(3,100)); ...
> plot3d(X[1],X[2],X[3],>anaglyph,>wire):
```



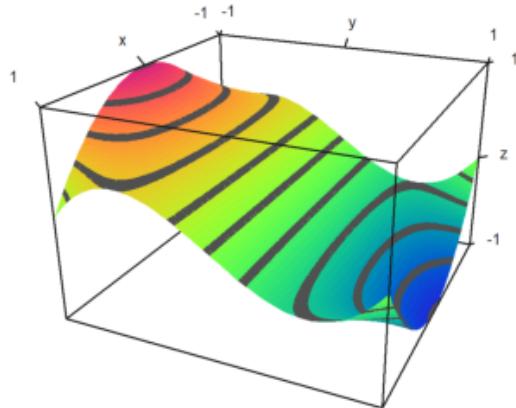
EMT juga dapat membuat plot dalam mode anaglyph. Untuk melihat plot seperti itu, Anda memerlukan kacamata berwarna merah/cyan.

```
> plot3d("x^2+y^3",>anaglyph,>contour,angle=30°) :
```



Seringkali skema warna spektral digunakan untuk plot. Ini menekankan ketinggian fungsinya.

```
>plot3d("x^2*y^3-y",>spectral,>contour,zoom=3.2) :
```

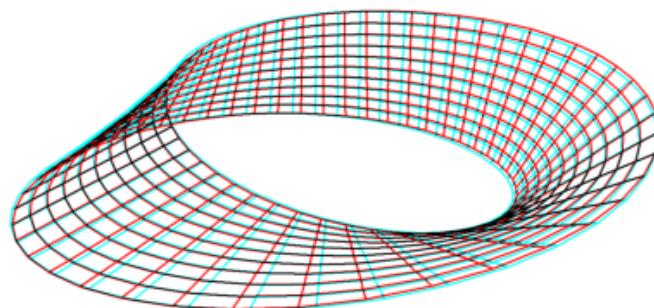


Euler juga dapat memplot permukaan yang diparameterisasi, jika parameternya adalah nilai x, y, dan z dari gambar kotak persegi panjang di ruang tersebut.

Untuk demo berikut, kami menyiapkan parameter u- dan v-, dan menghasilkan koordinat ruang dari parameter tersebut.

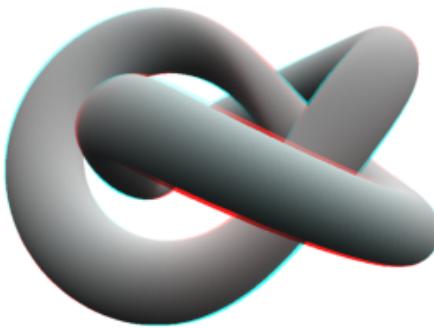
$t \leq 2\pi, \quad \text{quad } \frac{-\pi}{2} \leq s \leq \frac{\pi}{2}.$

```
>u=linspace(-1,1,10); v=linspace(0,2*pi,50)'; ...
>X=(3+u*cos(v/2))*cos(v); Y=(3+u*cos(v/2))*sin(v); Z=u*sin(v/2); ...
>plot3d(X,Y,Z,>anaglyph,<frame,>wire,scale=2.3);
```



Here is a more complicated example, which is majestic with red/cyan glasses.

```
>u:=linspace(-pi,pi,160); v:=linspace(-pi,pi,400)'; ...
>x:=(4*(1+.25*sin(3*v))+cos(u))*cos(2*v); ...
>y:=(4*(1+.25*sin(3*v))+cos(u))*sin(2*v); ...
>z=sin(u)+2*cos(3*v); ...
>plot3d(x,y,z,frame=0,scale=1.5,hue=1,light=[1,0,-1],zoom=2.8,>anaglyph):
```



Plot Statistik

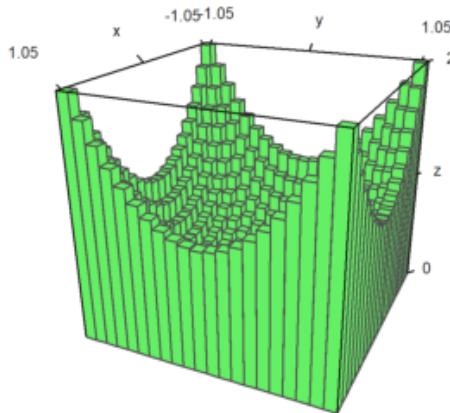
Plot batang juga dimungkinkan. Untuk ini, kita harus menyediakan

- x: row vector with n+1 elements
- y: column vector with n+1 elements
- z: nxn matrix of values.

z can be larger, but only nxn values will be used.

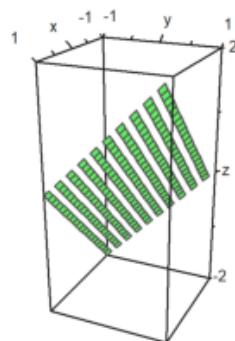
Dalam contoh ini, pertama-tama kita menghitung nilainya. Kemudian kita sesuaikan x dan y, sehingga vektor-vektornya berpusat pada nilai yang digunakan.

```
>x=-1:0.1:1; y=x'; z=x^2+y^2; ...
>xa=(x|1.1)-0.05; ya=(y_1.1)-0.05; ...
>plot3d(xa, ya, z, bar=true):
```



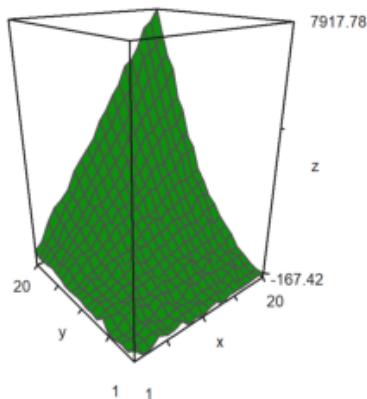
Dimungkinkan untuk membagi plot suatu permukaan menjadi dua bagian atau lebih.

```
>x=-1:0.1:1; y=x'; z=x+y; d=zeros(size(x)); ...
>plot3d(x,y,z,disconnect=2:2:20);
```

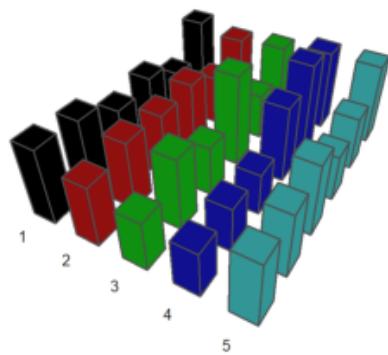


Jika memuat atau menghasilkan matriks data M dari file dan perlu memplotnya dalam 3D, Anda dapat menskalakan matriks ke [-1,1] dengan skala(M), atau menskalakan matriks dengan >zscale. Hal ini dapat dikombinasikan dengan faktor penskalaan individual yang ditetapkan sebagai tambahan.

```
>i=1:20; j=i'; ...
>plot3d(i*j^2+100*normal(20,20),>zscale,scale=[1,1,1.5],angle=-40°,zoom=1.8)
```

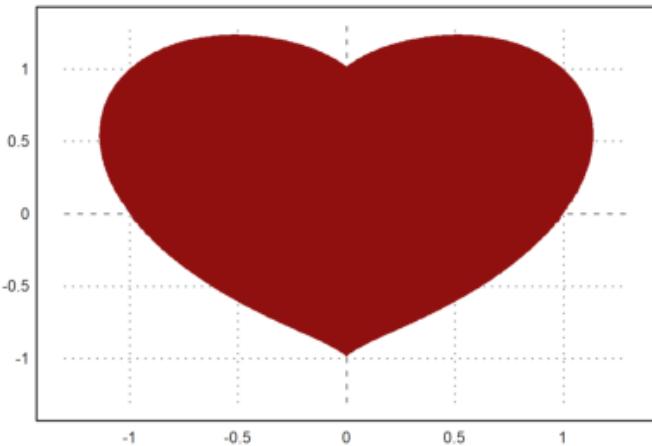


```
>Z=intrandom(5,100,6); v=zeros(5,6); ...
>loop 1 to 5; v[#]=getmultiplicities(1:6,Z[#]); end; ...
>columnsplot3d(v',scols=1:5,ccols=[1:5]):
```



Permukaan Benda Putar

```
>plot2d("(x^2+y^2-1)^3-x^2*y^3",r=1.3, ...
>style="#",color=red,<outline, ...
>level=[-2;0],n=100):
```



```
>ekspressi &= (x^2+y^2-1)^3-x^2*y^3; $ekspressi
```

$$(y^2 + x^2 - 1)^3 - x^2 y^3$$

Kami ingin memutar kurva hati di sekitar sumbu y. Inilah ungkapan yang mendefinisikan hati:

$$f(x, y) = (x^2 + y^2 - 1)^3 - x^2 \cdot y^3.$$

Next we set

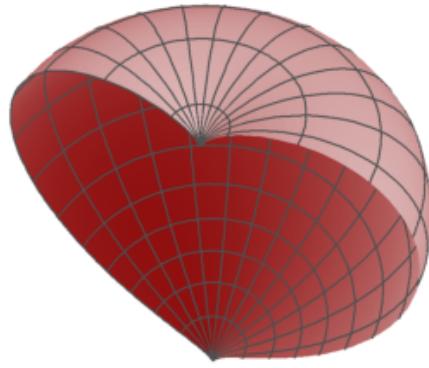
$$x = r \cdot \cos(a), \quad y = r \cdot \sin(a).$$

```
>function fr(r,a) &= ekspressi with [x=r*cos(a),y=r*sin(a)] | trigreduce; $f
```

$$(r^2 - 1)^3 + \frac{(\sin(5a) - \sin(3a) - 2 \sin a) r^5}{16}$$

Hal ini memungkinkan untuk mendefinisikan fungsi numerik, yang menyelesaikan r, jika a diberikan. Dengan fungsi tersebut kita dapat memplot jantung yang diputar sebagai permukaan parametrik.

```
>function map f(a) := bisect("fr",0,2;a); ...
>t=linspace(-pi/2,pi/2,100); r=f(t); ...
>s=linspace(pi,2pi,100)';
>plot3d(r*cos(t)*sin(s),r*cos(t)*cos(s),r*sin(t), ...
>>hue,<frame,color=red,zoom=4,amb=0,max=0.7,grid=12,height=50°):
```

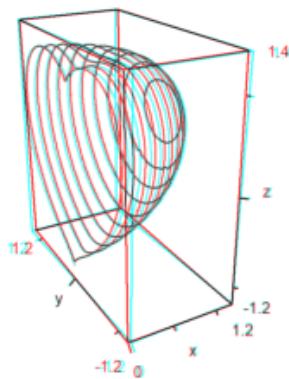


Berikut ini adalah plot 3D dari gambar di atas yang diputar mengelilingi sumbu z. Kami mendefinisikan fungsi yang mendeskripsikan objek.

```
>function f(x,y,z) ...
```

```
r=x^2+y^2;
return (r+z^2-1)^3-r*z^3;
endfunction
```

```
>plot3d("f(x,y,z)", ...
>xmin=0,xmax=1.2,ymin=-1.2,ymax=1.2,zmin=-1.2,zmax=1.4, ...
>implicit=1,angle=-30°,zoom=2.5,n=[10,100,60],>anaglyph):
```



Plot 3D Khusus

Fungsi plot3d bagus untuk dimiliki, tetapi tidak memenuhi semua kebutuhan. Selain rutinitas yang lebih mendasar, dimungkinkan untuk mendapatkan plot berbingkai dari objek apa pun yang Anda suka.

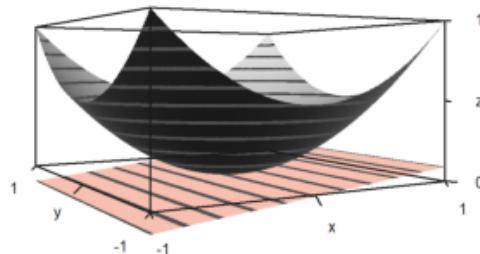
Meskipun Euler bukan program 3D, ia dapat menggabungkan beberapa objek dasar. Kami mencoba memvisualisasikan paraboloid dan garis singgungnya.

```
>function myplot ...
```

```
y=-1:0.01:1; x=(-1:0.01:1)';
plot3d(x,y,0.2*(x-0.1)/2,<scale,<frame,>hue, ...
hues=0.5,>contour,color=orange);
h=holding(1);
plot3d(x,y,(x^2+y^2)/2,<scale,<frame,>contour,>hue);
holding(h);
endfunction
```

Now framedplot() provides the frames, and sets the views.

```
>framedplot ("myplot", [-1,1,-1,1,0,1],height=0,angle=-30°, ...
> center=[0,0,-0.7],zoom=3):
```



Dengan cara yang sama, Anda dapat memplot bidang kontur secara manual. Perhatikan bahwa plot3d() menyetel jendela ke fullwindow(), secara default, tetapi plotcontourplane() berasumsi demikian.

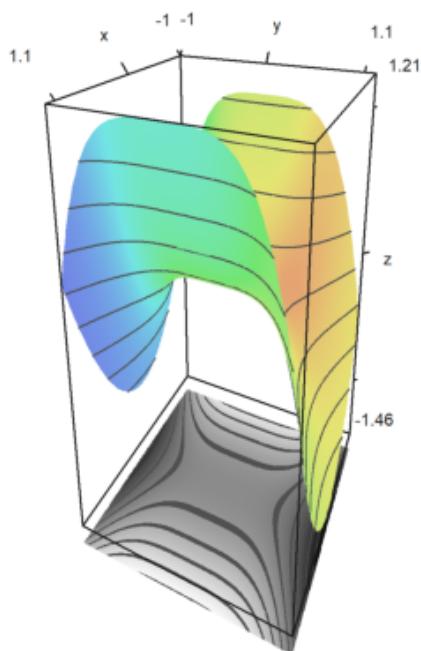
```
>x=-1:0.02:1.1; y=x'; z=x^2-y^4;
>function myplot (x,y,z) ...
```

```

zoom(2);
wi=fullwindow();
plotcontourplane(x,y,z,level="auto",<scale);
plot3d(x,y,z,>hue,<scale,>add,color=white,level="thin");
window(wi);
reset();
endfunction

```

```
>myplot(x,y,z):
```



Animasi

Euler dapat menggunakan frame untuk melakukan pra-komputasi animasi.

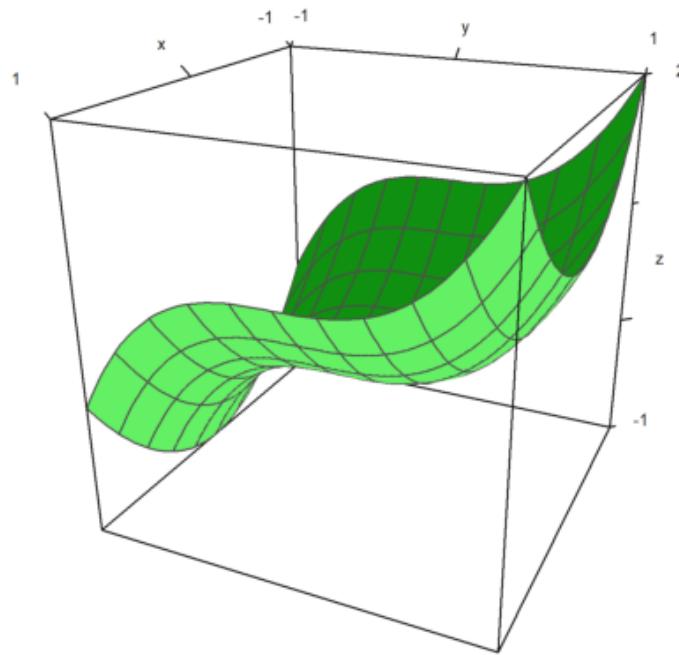
alah satu fungsi yang memanfaatkan teknik ini adalah memutar. Itu dapat mengubah sudut pandang dan menggambar ulang plot 3D. Fungsi ini memanggil addpage() untuk setiap plot baru. Akhirnya ia menganimasikan plotnya.

Silakan pelajari sumber rotasi untuk melihat lebih detail.

```

>function testplot () := plot3d("x^2+y^3"); ...
>rotate("testplot"); testplot():

```



Menggambar Povray

Dengan bantuan file Euler povray.e, Euler dapat menghasilkan file Povray. Hasilnya sangat bagus untuk dilihat.

Anda perlu menginstal Povray (32bit atau 64bit) dari <http://www.povray.org/>, dan meletakkan sub-direktori "bin" Povray ke jalur lingkungan, atau mengatur variabel "default-povray" dengan jalur lengkap yang mengarah ke "pvengine.exe".

ntarmuka Povray Euler menghasilkan file Povray di direktori home pengguna, dan memanggil Povray untuk menguraikan file-file ini. Nama file default adalah current.pov, dan direktori default adalah eulerhome(), biasanya c:\Users\Username\Euler. Povray menghasilkan file PNG, yang dapat dimuat oleh Euler ke dalam notebook. Untuk membersihkan file-file ini, gunakan povclear().

fungsi pov3d memiliki semangat yang sama dengan plot3d. Ini dapat menghasilkan grafik fungsi $f(x,y)$, atau permukaan dengan koordinat X,Y,Z dalam matriks, termasuk garis level opsional. Fungsi ini memulai raytracer secara otomatis, dan memuat adegan ke dalam notebook Euler.

elain pov3d(), ada banyak fungsi yang menghasilkan objek Povray. Fungsi-fungsi ini mengembalikan string, yang berisi kode Povray untuk objek. Untuk menggunakan fungsi ini, mulai file Povray dengan povstart(). Kemudian gunakan writeln(...) untuk menulis objek ke file adegan. Terakhir, akhiri file dengan povend(). Secara default, raytracer akan dimulai, dan PNG akan dimasukkan ke dalam notebook Euler.

Fungsi objek memiliki parameter yang disebut "tampilan", yang memerlukan string dengan kode Povray untuk tekstur dan penyelesaian objek. Fungsi povlook() dapat digunakan untuk menghasilkan string ini. Ini memiliki parameter untuk warna, transparansi, Phong Shading dll.

Perhatikan bahwa alam semesta Povray memiliki sistem koordinat lain. Antarmuka ini menerjemahkan semua koordinat ke sistem Povray. Jadi Anda dapat terus berpikir dalam sistem koordinat Euler dengan z menunjuk vertikal ke atas, dan sumbu x,y,z di tangan kanan. Fungsi pov3d memiliki semangat yang sama dengan plot3d. Ini dapat menghasilkan grafik fungsi $f(x,y)$, atau permukaan dengan koordinat X,Y,Z dalam matriks, termasuk garis level opsional. Fungsi ini memulai raytracer secara otomatis, dan memuat adegan ke dalam notebook Euler.

Anda perlu memuat file povray

```
>load povray;
```

Pastikan, direktori Povray bin ada di jalurnya. Jika tidak, edit variabel berikut sehingga berisi jalur ke povray yang dapat dieksekusi.

```
>defaultpovray="C:\Program Files\POV-Ray\v3.7\bin\pvengine.exe"
```

C:\Program Files\POV-Ray\v3.7\bin\pvengine.exe

Untuk kesan pertama, kami memplot fungsi sederhana. Perintah berikut menghasilkan file povray di direktori pengguna Anda, dan menjalankan Povray untuk penelusuran sinar file ini.

Jika Anda memulai perintah berikut, GUI Povray akan terbuka, menjalankan file, dan menutup secara otomatis. Karena alasan keamanan, Anda akan ditanya apakah Anda ingin mengizinkan file exe dijalankan. Anda dapat menekan batal untuk menghentikan pertanyaan lebih lanjut. Anda mungkin harus menekan OK di jendela Povray untuk mengonfirmasi dialog pengaktifan Povray.

```
>plot3d("x^2+y^2",zoom=2);
>pov3d("x^2+y^2",zoom=3);
```

Kita dapat membuat fungsinya transparan dan menambahkan penyelesaian lainnya. Kita juga dapat menambahkan garis level ke plot fungsi.

```
>pov3d("x^2+y^3",axiscolor=red,angle=-45°,>anaglyph, ...
> look=povlook(cyan,0.2),level=-1:0.5:1,zoom=3.8);
```

Terkadang perlu untuk mencegah penskalaan fungsi, dan menskalakan fungsi secara manual.

Kita memplot himpunan titik pada bidang kompleks, dimana hasil kali jarak ke 1 dan -1 sama dengan 1.

```
>pov3d("((x-1)^2+y^2)*((x+1)^2+y^2)/40",r=2, ...
> angle=-120°,level=1/40,dlevel=0.005,light=[-1,1,1],height=10°,n=50, ...
> <fscale,zoom=3.8);
```

Merencanakan dengan Koordinat

Daripada menggunakan fungsi, kita bisa memplotnya dengan koordinat. Seperti di plot3d, kita memerlukan tiga matriks untuk mendefinisikan objek.

Dalam contoh ini kita memutar suatu fungsi di sekitar sumbu z.

```
>function f(x) := x^3-x+1; ...
>x=-1:0.01:1; t=linspace(0,2pi,50)'; ...
>Z=x; X=cos(t)*f(x); Y=sin(t)*f(x); ...
>pov3d(X,Y,Z,angle=40°,look=povlook(red,0.1),height=50°,axis=0,zoom=4,light
```

```
Function povlook not found.
Try list ... to find functions!
Error in:
... f(x); pov3d(X,Y,Z,angle=40°,look=povlook(red,0.1),height=50°,a ...
^
```

Pada contoh berikut, kita memplot gelombang teredam. Kami menghasilkan gelombang dengan bahasa matriks Euler.

Kami juga menunjukkan, bagaimana objek tambahan dapat ditambahkan ke adegan pov3d. Untuk pembuatan objek, lihat contoh berikut. Perhatikan bahwa plot3d menskalakan plot, sehingga cocok dengan kubus satuan.

```
>r=linspace(0,1,80); phi=linspace(0,2pi,80)'; ...
>x=r*cos(phi); y=r*sin(phi); z=exp(-5*r)*cos(8*pi*r)/3; ...
>pov3d(x,y,z,zoom=6,axis=0,height=30°,add=povsphere([0.5,0,0.25],0.15,povlo
> w=500,h=300);
```

```
exec:
    return _exec(program,param,dir,print,hidden,wait);
povray:
    exec(program,params,defaulthome);
Try "trace errors" to inspect local variables after errors.
```

```
pov3d:  
if povray then povray(currentfile,w,h,w/h); endif;
```

Dengan metode peneduh canggih Povray, sangat sedikit titik yang dapat menghasilkan permukaan yang sangat halus. Hanya pada batas-batas dan dalam bayangan, triknya mungkin terlihat jelas.

Untuk ini, kita perlu menjumlahkan vektor normal di setiap titik matriks.

```
>Z &= x^2*y^3
```

$$\begin{matrix} 2 & 3 \\ x & y \end{matrix}$$

Persamaan permukaannya adalah $[x,y,Z]$. Kami menghitung dua turunan dari x dan y dan mengambil perkalian silangnya sebagai normal.

```
>dx &= diff([x,y,Z],x); dy &= diff([x,y,Z],y);
```

Kami mendefinisikan normal sebagai produk silang dari turunan ini, dan mendefinisikan fungsi koordinat

```
>N &= crossproduct(dx,dy); NX &= N[1]; NY &= N[2]; NZ &= N[3]; N,
```

$$\begin{bmatrix} 3 & 2 & 2 \\ -2x^y & -3x^y & 1 \end{bmatrix}$$

We use only 25 points.

```
>x=-1:0.5:1; y=x';  
>pov3d(x,y,Z(x,y),angle=10°, ...  
> xv=NX(x,y), yv=NY(x,y), zv=NZ(x,y), <shadow>;
```

The following is the Trefoil knot done by A. Busser in Povray. There is an improved version of this in the examples.

See: Examples\Trefoil Knot | Trefoil Knot

For a good look with not too many points, we add normal vectors here. We use Maxima to compute the normals for us. First, the three functions for the coordinates as symbolic expressions.

```
>X &= ((4+sin(3*y))+cos(x))*cos(2*y); ...
>Y &= ((4+sin(3*y))+cos(x))*sin(2*y); ...
>Z &= sin(x)+2*cos(3*y);
```

Then the two derivative vectors to x and y.

```
>dx &= diff([X,Y,Z],x); dy &= diff([X,Y,Z],y);
```

Now the normal, which is the cross product of the two derivatives.

```
>dn &= crossproduct(dx,dy);
```

We now evaluate all this numerically.

```
>x:=linspace(-%pi,%pi,40); y:=linspace(-%pi,%pi,100)';
```

The normal vectors are evaluations of the symbolic expressions dn[i] for i=1,2,3. The syntax for this is &"expression"(parameters). This is an alternative to the method in the previous example, where we defined symbolic expressions NX, NY, NZ first.

```
>pov3d(X(x,y),Y(x,y),Z(x,y),>anaglyph,axis=0,zoom=5,w=450,h=350, ...
> <shadow,look=povlook(blue), ...
> xv=&"dn[1]"(x,y), yv=&"dn[2]"(x,y), zv=&"dn[3]"(x,y));
```

We can also generate a grid in 3D.

```
>povstart(zoom=4); ...
>x=-1:0.5:1; r=1-(x+1)^2/6; ...
>t=(0°:30°:360°)'; y=r*cos(t); z=r*sin(t); ...
>writeln(povgrid(x,y,z,d=0.02,dballs=0.05)); ...
>povend();
```

```
exec:
    return _exec(program,param,dir,print,hidden,wait);
povray:
    exec(program,params,defaulthome);
Try "trace errors" to inspect local variables after errors.
povend:
    povray(file,w,h,aspect,exit);
```

With `povgrid()`, curves are possible.

```
>povstart(center=[0,0,1],zoom=3.6); ...
>t=linspace(0,2,1000); r=exp(-t); ...
>x=cos(2*pi*10*t)*r; y=sin(2*pi*10*t)*r; z=t; ...
>writeln(povgrid(x,y,z,povlook(red))); ...
>writeAxis(0,2,axis=3); ...
>povend();
```

Objek Povray

Di atas, kami menggunakan `pov3d` untuk memplot permukaan. Antarmuka `povray` di Euler juga dapat menghasilkan objek Povray. Objek ini disimpan sebagai string di Euler, dan perlu ditulis ke file Povray.

Kami memulai output dengan `povstart()`.

```
>povstart(zoom=4);
```

Pertama kita mendefinisikan tiga silinder, dan menyimpannya dalam string di Euler.

Fungsi `povx()` dll. hanya mengembalikan vektor [1,0,0], yang dapat digunakan sebagai gantinya.

```
>c1=povcylinder(-povx,povx,1,povlook(red)); ...
>c2=povcylinder(-povy,povy,1,povlook(yellow)); ...
>c3=povcylinder(-povz,povz,1,povlook(blue)); ...
```

String tersebut berisi kode Povray, yang tidak perlu kita pahami pada saat itu.

Fungsi povx() dll. hanya mengembalikan vektor [1,0,0], yang dapat digunakan sebagai gantinya.

```
>c2
```

```
cylinder { <0,0,-1>, <0,0,1>, 1
texture { pigment { color rgb <0.941176,0.941176,0.392157> } }
finish { ambient 0.2 }
}
```

As you see, we added texture to the objects in three different colors.

Hal ini dilakukan oleh povlook(), yang mengembalikan string dengan kode Povray yang relevan. Kita dapat menggunakan warna default Euler, atau menentukan warna kita sendiri. Kita juga dapat menambahkan transparansi, atau mengubah cahaya sekitar.

```
>povlook(rgb(0.1,0.2,0.3),0.1,0.5)
```

```
texture { pigment { color rgbf <0.101961,0.2,0.301961,0.1> } }
finish { ambient 0.5 }
```

Sekarang kita mendefinisikan objek persimpangan, dan menulis hasilnya ke file. i dilakukan oleh povlook(), yang mengembalikan string dengan kode Povray yang relevan. Kita dapat menggunakan warna default Euler, atau menentukan warna kita sendiri. Kita juga dapat menambahkan transparansi, atau mengubah cahaya sekitar.

```
>writeln(povintersection([c1,c2,c3]));
```

Persimpangan tiga silinder sulit untuk divisualisasikan jika Anda belum pernah melihatnya sebelumnya.

```
>povend;
```

Fungsi berikut menghasilkan fraktal secara rekursif.

Fungsi pertama menunjukkan bagaimana Euler menangani objek Povray sederhana. Fungsi povbox() mengembalikan string, yang berisi koordinat kotak, tekstur, dan hasil akhir.

```
>function onebox(x,y,z,d) := povbox([x,y,z],[x+d,y+d,z+d],povlook());  
>function fractal (x,y,z,h,n) ...
```

```
if n==1 then writeln(onebox(x,y,z,h));  
else  
    h=h/3;  
    fractal(x,y,z,h,n-1);  
    fractal(x+2*h,y,z,h,n-1);  
    fractal(x,y+2*h,z,h,n-1);  
    fractal(x,y,z+2*h,h,n-1);  
    fractal(x+2*h,y+2*h,z,h,n-1);  
    fractal(x+2*h,y,z+2*h,h,n-1);  
    fractal(x,y+2*h,z+2*h,h,n-1);  
    fractal(x+2*h,y+2*h,z+2*h,h,n-1);  
    fractal(x+h,y+h,z+h,h,n-1);  
endif;  
endfunction
```

```
>povstart(fade=10,<shadow);  
>fractal(-1,-1,-1,2,4);  
>povend();
```

Perbedaan memungkinkan pemisahan satu objek dari objek lainnya. Seperti persimpangan, ada bagian dari objek CSG di Povray.

```
>povstart(light=[5,-5,5],fade=10);
```

For this demonstration, we define an object in Povray, instead of using a string in Euler. Definitions are written to the file immediately.

A box coordinate of -1 just means [-1,-1,-1].

```
>povdefine ("mycube",povbox (-1,1));
```

We can use this object in povobject(), which returns a string as usual.

```
>c1=povobject ("mycube", povlook (red)) ;
```

We generate a second cube, and rotate and scale it a bit.

```
>c2=povobject ("mycube", povlook (yellow), translate=[1,1,1], ...
>    rotate=xrotate(10°)+yrotate(10°), scale=1.2);
```

Then we take the difference of the two objects.

```
>writeln (povdifference (c1, c2));
```

Now add three axes.

```
>writeAxis (-1.2,1.2, axis=1); ...
>writeAxis (-1.2,1.2, axis=2); ...
>writeAxis (-1.2,1.2, axis=4); ...
>povend();
```

Fungsi Implisit

Povray dapat memplot himpunan di mana $f(x,y,z)=0$, seperti parameter implisit di plot3d. Namun hasilnya terlihat jauh lebih baik.

intaks untuk fungsinya sedikit berbeda. Anda tidak dapat menggunakan keluaran ekspresi Maxima atau Euler.

Latex: $((x^2+y^2-c^2)^2+(z^2-1)^2)*((y^2+z^2-c^2)^2+(x^2-1)^2)*((z^2+x^2-c^2)^2+(y^2-1)^2)=d$

```
>povstart (angle=70°, height=50°, zoom=4);
>c=0.1; d=0.1; ...
>writeln (povsurface (" (pow (pow (x, 2) +pow (y, 2) -pow (c, 2) , 2 ) +pow (pow (z, 2) -1, 2 )) *
```

Error : Povray error!

Error generated by error() command

```
povray:
    error("Povray error!");
```

Try "trace errors" to inspect local variables after errors.

povend:

```
povray(file,w,h,aspect,exit);
```

```
>povstart(angle=25°,height=10°);
>writeln(povsurface("pow(x,2)+pow(y,2)*pow(z,2)-1",povlook(blue),povbox(-2,
>povend();
>povstart(angle=70°,height=50°,zoom=4);
```

Create the implicit surface. Note the different syntax in the expression.

```
>writeln(povsurface("pow(x,2)*y-pow(y,3)-pow(z,2)",povlook(green))); ...
>writeAxes(); ...
>povend();
```

Objek Jaring

Dalam contoh ini, kami menunjukkan cara membuat objek mesh, dan menggambarnya dengan informasi tambahan.

Kita ingin memaksimalkan xy pada kondisi $x+y=1$ dan mendemonstrasikan sentuhan tangensial garis datar.

```
>povstart(angle=-10°,center=[0.5,0.5,0.5],zoom=7);
```

We cannot store the object in a string as before, since it is too large. So we define the object in a Povray file using declare. The function povtriangle() does this automatically. It can accept normal vectors just like pov3d().

The following defines the mesh object, and writes it immediately into the file.

```
>x=0:0.02:1; y=x'; z=x*y; vx=-y; vy=-x; vz=1;
>mesh=povtriangles(x,y,z,"",vx,vy,vz);
```

Now we define two discs, which will be intersected with the surface.

```
>cl=povdisc([0.5,0.5,0],[1,1,0],2); ...
>ll=povdisc([0,0,1/4],[0,0,1],2);
```

Write the surface minus the two discs.

```
>writeln(povdifference(mesh,povunion([cl,ll]),povlook(green)));
```

Write the two intersections.

```
>writeln(povintersection([mesh,cl],povlook(red))); ...
>writeln(povintersection([mesh,ll],povlook(gray)));
```

Write a point at the maximum.

```
>writeln(povpoint([1/2,1/2,1/4],povlook(gray),size=2*defaultpointsize));
```

Add axes and finish.

```
>writeAxes(0,1,0,1,0,1,d=0.015); ...
>povend();
```

Anaglyphs di Povray

Untuk menghasilkan anaglyph untuk kacamata merah/cyan, Povray harus dijalankan dua kali dari posisi kamera berbeda. Ini menghasilkan dua file Povray dan dua file PNG, yang dimuat dengan fungsi loadanaglyph().

Untuk melihat contoh berikut dengan benar.

Fungsi pov3d() memiliki saklar sederhana untuk menghasilkan anaglyph.

```
>pov3d("-exp(-x^2-y^2)/2",r=2,height=45°,>anaglyph, ...
> center=[0,0,0.5],zoom=3.5);
```

If you create a scene with objects, you need to put the generation of the scene into a function, and run it twice with different values for the anaglyph parameter.

```
>function myscene ...
s=povsphere(povc,1);
cl=povcylinder(-povz,povz,0.5);
clx=povobject(cl,rotate=xrotate(90°));
cly=povobject(cl,rotate=yrotate(90°));
c=povbox([-1,-1,0],1);
un=povunion([cl,clx,cly,c]);
obj=povdifference(s,un,povlook(red));
writeln(obj);
writeAxes();
endfunction
```

The function povanaglyph() does all this. The parameters are like in povstart() and povend() combined.

```
>povanaglyph ("myscene", zoom=4.5);
```

Mendefinisikan Objek sendiri

ntarmuka povray Euler berisi banyak objek. Namun Anda tidak dibatasi pada hal ini. Anda dapat membuat objek sendiri, yang menggabungkan objek lain, atau merupakan objek yang benar-benar baru.

Kami mendemonstrasikan torus. Perintah Povray untuk ini adalah "torus". Jadi kami mengembalikan string dengan perintah ini dan parameternya. Perhatikan bahwa torus selalu berpusat pada titik asal.

```
>function povdonat (r1,r2,look="") ...
return "torus {" + r1 + "," + r2 + look + "}";
endfunction
```

Here is our first torus.

```
>t1=povdonat (0.8,0.2)
```

```
torus {0.8,0.2}
```

Let us use this object to create a second torus, translated and rotated.

```
>t2=povobject(t1,rotate=xrotate(90°),translate=[0.8,0,0])
```

```
object { torus { 0.8,0.2}
  rotate 90 *x
  translate <0.8,0,0>
}
```

Now we place these objects into a scene. For the look, we use Phong Shading.

```
>povstart(center=[0.4,0,0],angle=0°,zoom=3.8,aspect=1.5); ...
>writeln(povobject(t1,povlook(green,phong=1))); ...
>writeln(povobject(t2,povlook(green,phong=1))); ...
```

```
>povend();
```

calls the Povray program. However, in case of errors, it does not display the error. You should therefore use

```
>povend(<exit);
```

if anything did not work. This will leave the Povray window open.

```
>povend(h=320,w=480);
```

```
Function povstart not found.
Try list ... to find functions!
Error in:
povstart(center=[0.4,0,0],angle=0°,zoom=3.8,aspect=1.5); writeln(povobjec
```

Here is a more elaborate example. We solve

$$Ax \leq b, \quad x \geq 0, \quad c.x \rightarrow \text{Max.}$$

and show the feasible points and the optimum in a 3D plot.

```
>A=[10,8,4;5,6,8;6,3,2;9,5,6];
>b=[10,10,10,10]';
>c=[1,1,1];
```

First, let us check, if this example has a solution at all.

```
>x=simplex(A,b,c,>max,>check)'
```

```
[0, 1, 0.5]
```

Yes, it has.

Next we define two objects. The first is the plane

$$a \cdot x \leq b$$

```
>function oneplane (a,b,look=""') ...
```

```
    return povplane(a,b,look)
endfunction
```

Then we define the intersection of all half spaces and a cube.

```
>function adm (A, b, r, look=""') ...
```

```
ol=[];
loop 1 to rows(A); ol=ol|oneplane(A[#],b[#]); end;
ol=ol|povbox([0,0,0],[r,r,r]);
return povintersection(ol,look);
endfunction
```

We can now plot the scene.

```
>povstart(angle=120°,center=[0.5,0.5,0.5],zoom=3.5); ...
>writeln(adm(A,b,2,povlook(green,0.4))); ...
>writeAxes(0,1.3,0,1.6,0,1.5); ...
```

The following is a circle around the optimum.

```
>writeln(povintersection([povsphere(x,0.5),povplane(c,c.x')]), ...
>    povlook(red,0.9));
```

And an error in the direction of the optimum.

```
>writeln(povarrow(x,c*0.5,povlook(red)));
```

We add text to the screen. Text is just a 3D object. We need to place and turn it according to our view.

```
>writeln(povtext("Linear Problem", [0,0.2,1.3], size=0.05, rotate=5°)); ...
>povend();
```

Contoh Lainnya

Anda dapat menemukan beberapa contoh Povray di Euler di file berikut.

ee: Examples/Dandelin Spheres

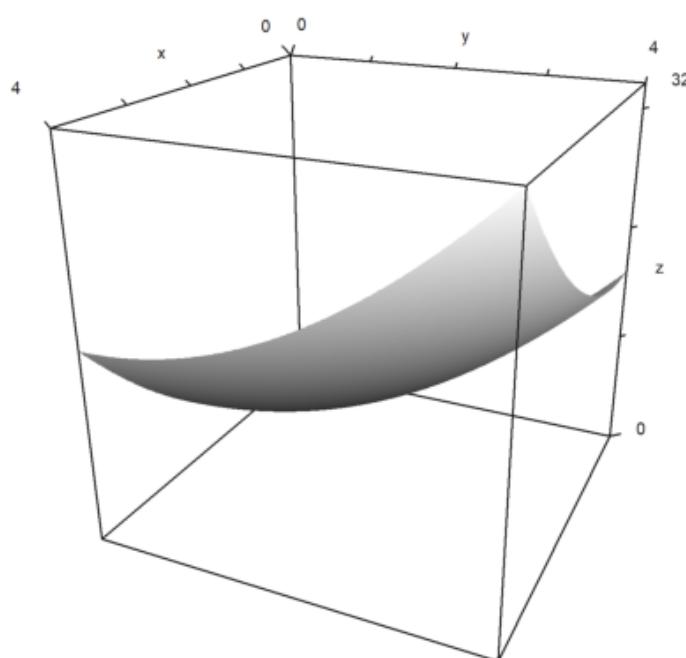
See: Examples/Donat Math

See: Examples/Trefoil Knot

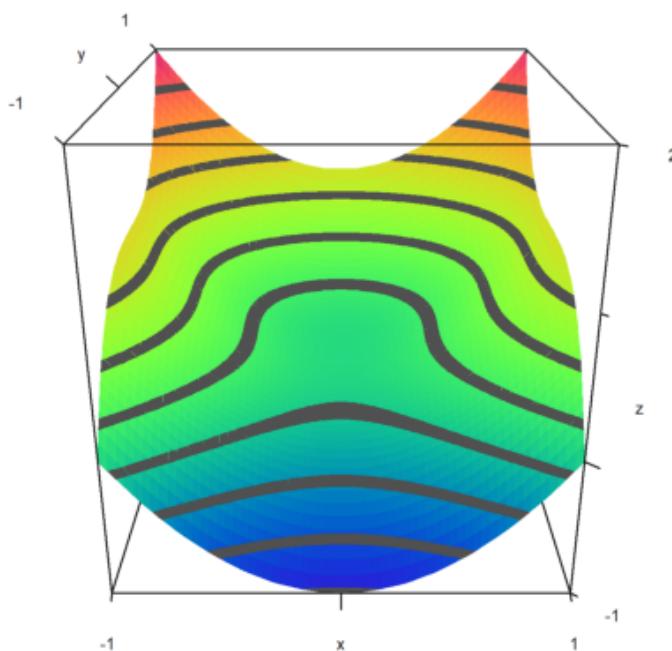
See: Examples/Optimization by Affine Scaling

Latihan Soal

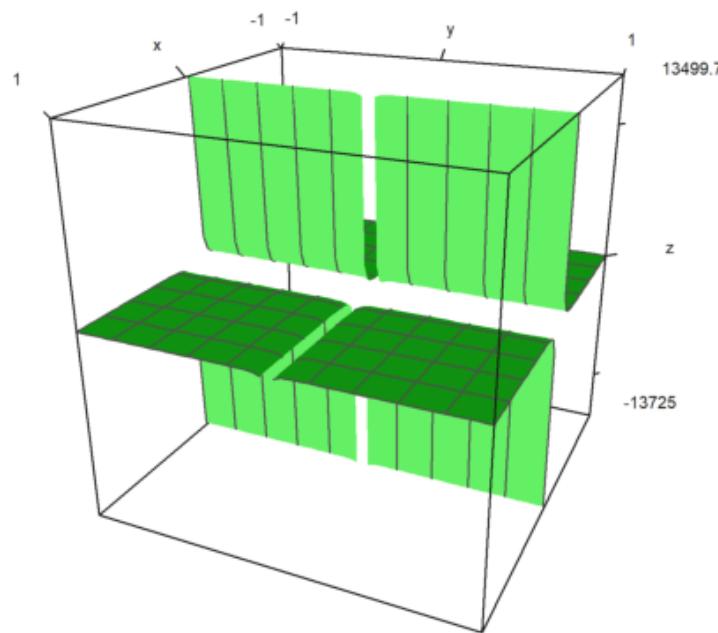
```
>plot3d("y^2+x^2", a=0, b=4, c=0, d=4, n=200, level=0, light=[1,1,0]):
```



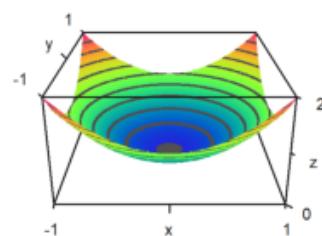
```
>plot3d("x^2+y^3",angle=0°,>contour,>spectral):
```



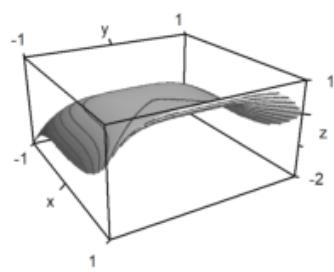
```
>plot3d("x-(1/2x^3)-(1/4y^2)",r=1,title="z=x-(1/2x^3)-(1/4y^2)":
```



```
>plot3d("y^2+x^2",angle=0°,>contour,>spectral):
```



```
>plot3d("x^3-y^4",>contour,angle=60°,level=-2:0.2:2):
```



BAB

Kalkulus dengan EMT

Materi Kalkulus mencakup di antaranya:

- Fungsi (fungsi aljabar, trigonometri, eksponensial, logaritma, komposisi fungsi)
- Limit Fungsi,
- Turunan Fungsi,
- Integral Tak Tentu,
- Integral Tentu dan Aplikasinya,
- Barisan dan Deret (kekonvergenan barisan dan deret).

EMT (bersama Maxima) dapat digunakan untuk melakukan semua perhitungan di dalam kalkulus, baik secara numerik maupun analitik (eksak).

Mendefinisikan Fungsi

Terdapat beberapa cara mendefinisikan fungsi pada EMT, yakni:

- Menggunakan format `nama_fungsi := rumus_fungsi` (untuk fungsi numerik),
- Menggunakan format `nama_fungsi &= rumus_fungsi` (untuk fungsi simbolik, namun dapat dihitung secara numerik),
- Menggunakan format `nama_fungsi &&= rumus_fungsi` (untuk fungsi simbolik murni, tidak dapat dihitung langsung),
- Fungsi sebagai program EMT.

Setiap format harus diawali dengan perintah `function` (bukan sebagai ekspresi).

Berikut adalah beberapa contoh cara mendefinisikan fungsi:

$$f(x) = 2x^2 + e^{\sin(x)}.$$

```
>function f(x) := 2*x^2+exp(sin(x)) // fungsi numerik  
>f(0), f(1), f(pi)
```

```
1
4.31977682472
20.7392088022
```

```
>f(a) // tidak dapat dihitung nilainya
```

Variable or function a not found.

Error in:

```
f(a) // tidak dapat dihitung nilainya ...
^
```

Silakan Anda plot kurva fungsi di atas!

Berikutnya kita definisikan fungsi:

$$g(x) = \frac{\sqrt{x^2 - 3x}}{x + 1}.$$

```
>function g(x) := sqrt(x^2-3*x) / (x+1)
>g(3)
```

0

```
>g(0)
```

0

```
>g(1) // kompleks, tidak dapat dihitung oleh fungsi numerik
```

Floating point error!

Error in sqrt

Try "trace errors" to inspect local variables after errors.

g:

```
useglobal; return sqrt(x^2-3*x) / (x+1)
```

Error in:

```
g(1) // kompleks, tidak dapat dihitung oleh fungsi numerik ...
^
```

Silakan Anda plot kurva fungsi di atas!

```
>f(g(5)) // komposisi fungsi
```

2.20920171961

```
>g(f(5))
```

0.950898070639

```
>function h(x) := f(g(x)) // definisi komposisi fungsi
>h(5) // sama dengan f(g(5))
```

2.20920171961

Silakan Anda plot kurva fungsi komposisi fungsi f dan g:

$$h(x) = f(g(x))$$

dan

$$u(x) = g(f(x))$$

bersama-sama kurva fungsi f dan g dalam satu bidang koordinat.

```
>f(0:10) // nilai-nilai f(0), f(1), f(2), ..., f(10)
```

[1, 4.31978, 10.4826, 19.1516, 32.4692, 50.3833, 72.7562,
99.929, 130.69, 163.51, 200.58]

```
>fmap(0:10) // sama dengan f(0:10), berlaku untuk semua fungsi
```

[1, 4.31978, 10.4826, 19.1516, 32.4692, 50.3833, 72.7562,
99.929, 130.69, 163.51, 200.58]

```
>gmap(200:210)
```

[0.987534, 0.987596, 0.987657, 0.987718, 0.987778, 0.987837,
0.987896, 0.987954, 0.988012, 0.988069, 0.988126]

Misalkan kita akan mendefinisikan fungsi

$$f(x) = \begin{cases} x^3 & x > 0 \\ x^2 & x \leq 0. \end{cases}$$

Fungsi tersebut tidak dapat didefinisikan sebagai fungsi numerik secara "inline" menggunakan format `:=`, melainkan didefinisikan sebagai program. Perhatikan, kata "map" digunakan agar fungsi dapat menerima vektor sebagai input, dan hasilnya berupa vektor. Jika tanpa kata "map" fungsinya hanya dapat menerima input satu nilai.

```
>function map f(x) ...
```

```
    if x>0 then return x^3
    else return x^2
    endif;
endfunction
```

```
>f(1)
```

1

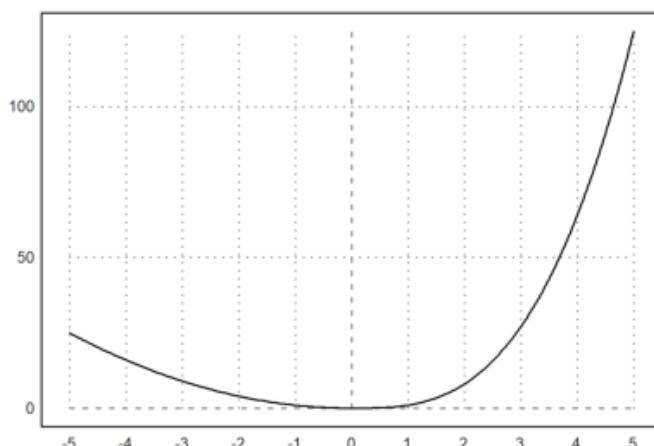
```
>f(-2)
```

4

```
>f(-5:5)
```

[25, 16, 9, 4, 1, 0, 1, 8, 27, 64, 125]

```
>aspect(1.5); plot2d("f(x)", -5, 5):
```



```
>function f(x) &= 2*x^2 // fungsi simbolik
```

$$\frac{x^2}{2}$$

```
>$f(a) // nilai fungsi secara simbolik
```

$$2e^a$$

```
>f(E) // nilai fungsi berupa bilangan desimal
```

30.308524483

```
>$f(E), $float(%)
```

$$2e^e$$

30.30852448295852

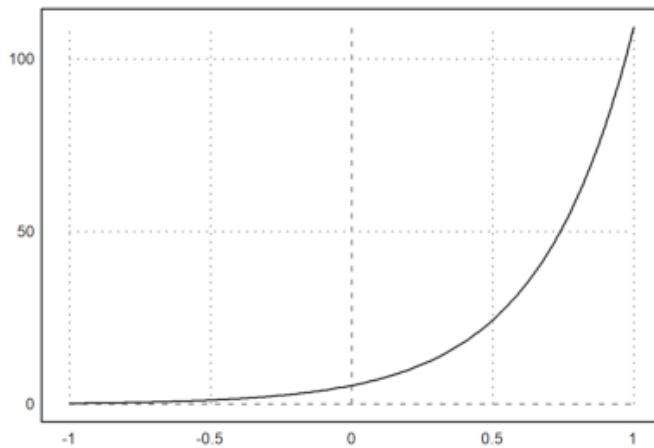
```
>function g(x) &= 3*x+1
```

$$3x + 1$$

```
>function h(x) &= f(g(x)) // komposisi fungsi
```

$$\frac{3x^3 + 1}{2e}$$

```
>plot2d("h(x)", -1, 1):
```



Latihan

Bukalah buku Kalkulus. Cari dan pilih beberapa (paling sedikit 5 fungsi berbeda tipe/bentuk/jenis) fungsi dari buku tersebut, kemudian definisikan fungsi-fungsi tersebut dan komposisinya di EMT pada baris-baris perintah berikut (jika perlu tambahkan lagi). Untuk setiap fungsi, hitung beberapa nilainya, baik untuk satu nilai maupun vektor. Gambar grafik fungsi-fungsi tersebut dan komposisi-komposisi 2 fungsi.

Juga, carilah fungsi beberapa (dua) variabel. Lakukan hal sama seperti di atas.

1

```
>function f(x) := -x^2-2*x+2
>function g(x) := 4^x+2
>f(1), f(-2), f(4), g(2), g(4), g(-1)
```

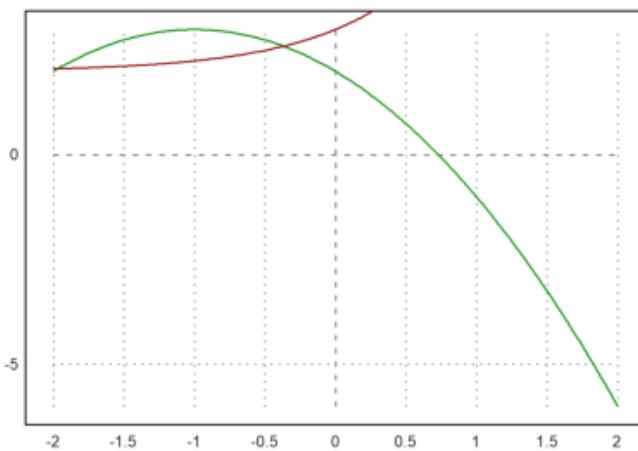
-1
2
-22
18
258
2.25

```
>f(3), f(-1), f(2), g(3), g(7), g(-1)
```

-13
3
-6
66
16386
2.25

3

```
>plot2d("f(x)", color=green), plot2d("g(x)", color=red, >add) :
```

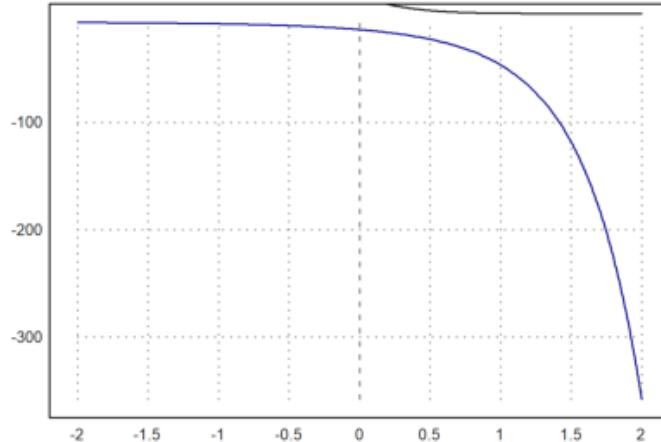


4

```
>function h(x) := f(g(x))
>function j(x) := g(f(x))
>h(1), h(-2), h(4), j(2), j(4), j(-1)
```

-46
-6.37890625
-67078
2.00024414063
2
66

```
>plot2d("h(x)", color=blue), plot2d("j(x)", >spectral, >add) :
```



Menghitung Limit

Perhitungan limit pada EMT dapat dilakukan dengan menggunakan fungsi Maxima, yakni "limit". Fungsi "limit" dapat digunakan untuk menghitung limit fungsi dalam bentuk ekspresi maupun fungsi yang sudah didefinisikan sebelumnya. Nilai limit dapat dihitung pada sebarang nilai atau pada tak hingga (-inf, minf, dan inf). Limit kiri dan limit kanan juga dapat dihitung, dengan cara memberi opsi "plus" atau "minus". Hasil limit dapat berupa nilai, "und" (tak definisi), "ind" (tak tentu namun terbatas), "infinity" (kompleks tak hingga).

Perhatikan beberapa contoh berikut. Perhatikan cara menampilkan perhitungan secara lengkap, tidak hanya menampilkan hasilnya saja.

```
>$showev('limit(sqrt(x^2-3*x)/(x+1),x,inf))
```

$$\lim_{x \rightarrow \infty} \frac{\sqrt{x^2 - 3x}}{x + 1} = 1$$

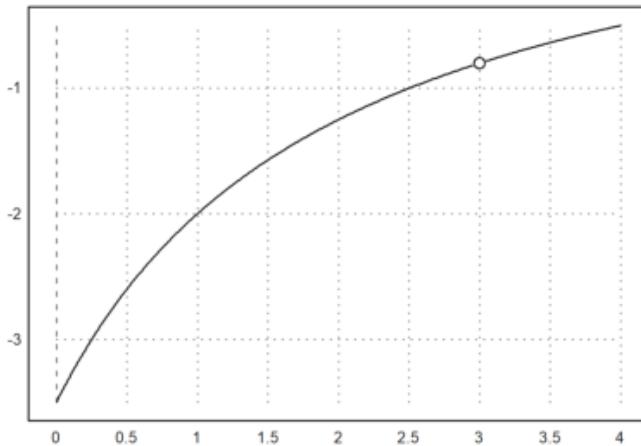
```
>$limit((x^3-13*x^2+51*x-63)/(x^3-4*x^2-3*x+18),x,3)
```

$$-\frac{4}{5}$$

maxima: 'limit((x^3-13*x^2+51*x-63)/(x^3-4*x^2-3*x+18),x,3)=limit((x^3-13*x^2+51*x-63)/(x^3-4*x^2-3*x+18),x,3)

Fungsi tersebut diskontinu di titik $x=3$. Berikut adalah grafik fungsinya.

```
>aspect(1.5); plot2d("(x^3-13*x^2+51*x-63)/(x^3-4*x^2-3*x+18)", 0, 4); plot2d
```



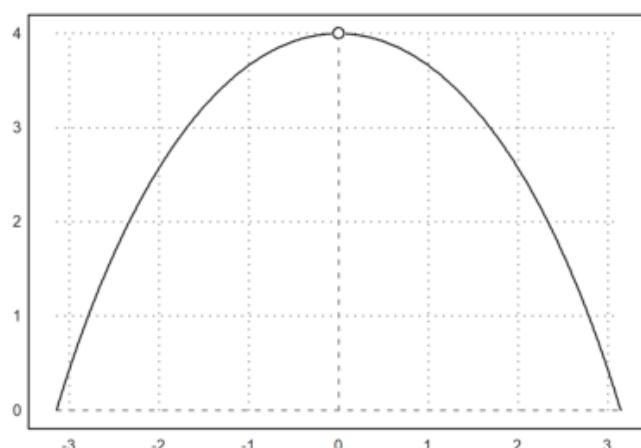
```
>$limit(2*x*sin(x)/(1-cos(x)), x, 0)
```

4

maxima: 'limit(2*x*sin(x)/(1-cos(x)),x,0)=limit(2*x*sin(x)/(1-cos(x)),x,0)

Fungsi tersebut diskontinu di titik $x=0$. Berikut adalah grafik fungsinya.

```
>plot2d("2*x*sin(x)/(1-cos(x))", -pi, pi); plot2d(0, 4, >points, style="ow", >add
```



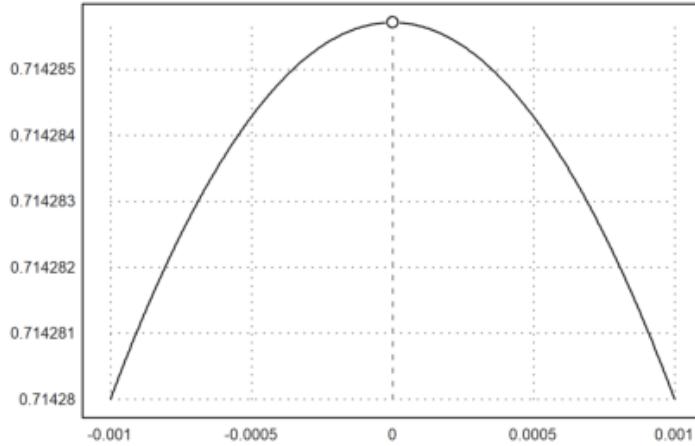
```
>$limit(cot(7*h)/cot(5*h), h, 0)
```

$$\frac{5}{7}$$

maxima: showev('limit(cot(7*h)/cot(5*h),h,0))

Fungsi tersebut juga diskontinu (karena tidak terdefinisi) di $x=0$. Berikut adalah grafiknya.

```
>plot2d("cot (7*x)/cot (5*x)", -0.001, 0.001); plot2d(0, 5/7, >points, style="ow",
```



```
>$showev('limit(((x/8)^(1/3)-1)/(x-8), x, 8))
```

$$\lim_{x \rightarrow 8} \frac{\frac{x^{\frac{1}{3}}}{2} - 1}{x - 8} = \frac{1}{24}$$

Tunjukkan limit tersebut dengan grafik, seperti contoh-contoh sebelumnya.

```
>$showev('limit(1/(2*x-1), x, 0))
```

$$\lim_{x \rightarrow 0} \frac{1}{2x - 1} = -1$$

Tunjukkan limit tersebut dengan grafik, seperti contoh-contoh sebelumnya.

```
>showev('limit((x^2-3*x-10)/(x-5),x,5))
```

$$\lim_{x \rightarrow 5} \frac{x^2 - 3x - 10}{x - 5} = 7$$

Tunjukkan limit tersebut dengan grafik, seperti contoh-contoh sebelumnya.

```
>showev('limit(sqrt(x^2+x)-x,x,inf))
```

$$\lim_{x \rightarrow \infty} \sqrt{x^2 + x} - x = \frac{1}{2}$$

Tunjukkan limit tersebut dengan grafik, seperti contoh-contoh sebelumnya.

```
>showev('limit(abs(x-1)/(x-1),x,1,minus))
```

$$\lim_{x \uparrow 1} \frac{|x - 1|}{x - 1} = -1$$

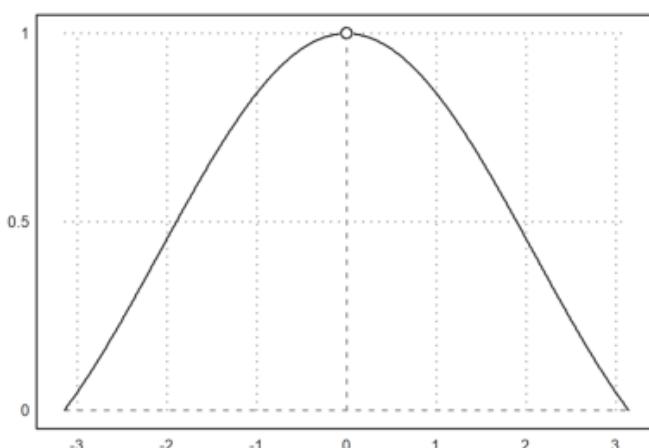
Hitung limit di atas untuk x menuju 1 dari kanan.

Tunjukkan limit tersebut dengan grafik, seperti contoh-contoh sebelumnya.

```
>showev('limit(sin(x)/x,x,0))
```

$$\lim_{x \rightarrow 0} \frac{\sin x}{x} = 1$$

```
>plot2d("sin(x)/x",-pi,pi); plot2d(0,1,>points,style="ow",>add):
```



```
>$showev('limit(sin(x^3)/x,x,0))
```

$$\lim_{x \rightarrow 0} \frac{\sin x^3}{x} = 0$$

Tunjukkan limit tersebut dengan grafik, seperti contoh-contoh sebelumnya.

```
>$showev('limit(log(x), x, minf))
```

$$\lim_{x \rightarrow -\infty} \log x = \text{infinity}$$

```
>$showev('limit((-2)^x,x, inf))
```

$$\lim_{x \rightarrow \infty} (-2)^x = \text{infinity}$$

```
>$showev('limit(t-sqrt(2-t),t,2,minus))
```

$$\lim_{t \uparrow 2} t - \sqrt{2-t} = 2$$

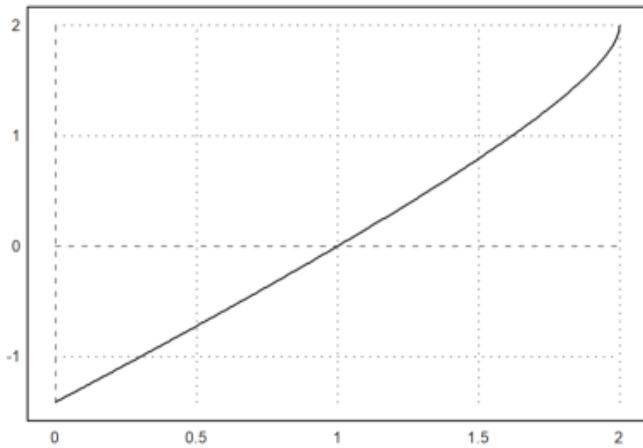
```
>$showev('limit(t-sqrt(2-t),t,2,plus))
```

$$\lim_{t \downarrow 2} t - \sqrt{2-t} = 2$$

```
>$showev('limit(t-sqrt(2-t),t,5,plus)) // Perhatikan hasilnya
```

$$\lim_{t \downarrow 5} t - \sqrt{2-t} = 5 - \sqrt{3} i$$

```
>plot2d("x-sqrt(2-x)",0,2):
```



```
>$showev('limit((x^2-9)/(2*x^2-5*x-3),x,3))
```

$$\lim_{x \rightarrow 3} \frac{x^2 - 9}{2x^2 - 5x - 3} = \frac{6}{7}$$

Tunjukkan limit tersebut dengan grafik, seperti contoh-contoh sebelumnya.

```
>$showev('limit((1-cos(x))/x,x,0))
```

$$\lim_{x \rightarrow 0} \frac{1 - \cos x}{x} = 0$$

Tunjukkan limit tersebut dengan grafik, seperti contoh-contoh sebelumnya.

```
>$showev('limit((x^2+abs(x))/(x^2-abs(x)),x,0))
```

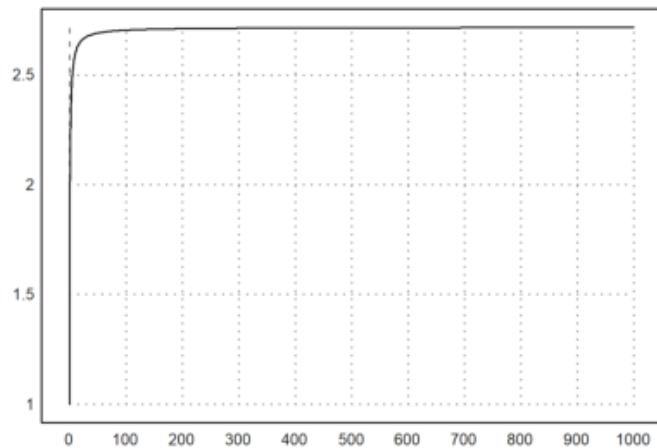
$$\lim_{x \rightarrow 0} \frac{|x| + x^2}{x^2 - |x|} = -1$$

Tunjukkan limit tersebut dengan grafik, seperti contoh-contoh sebelumnya.

```
>$showev('limit((1+1/x)^x,x,inf))
```

$$\lim_{x \rightarrow \infty} \left(\frac{1}{x} + 1 \right)^x = e$$

```
>plot2d("(1+1/x)^x", 0, 1000) :
```



```
>$showev('limit((1+k/x)^x, x, inf))
```

$$\lim_{x \rightarrow \infty} \left(\frac{k}{x} + 1 \right)^x = e^k$$

```
>$showev('limit((1+x)^(1/x), x, 0))
```

$$\lim_{x \rightarrow 0} (x + 1)^{\frac{1}{x}} = e$$

Tunjukkan limit tersebut dengan grafik, seperti contoh-contoh sebelumnya.

```
>$showev('limit((x/(x+k))^x, x, inf))
```

$$\lim_{x \rightarrow \infty} \left(\frac{x}{x + k} \right)^x = e^{-k}$$

```
>$showev('limit((E^x-E^2)/(x-2), x, 2))
```

$$\lim_{x \rightarrow 2} \frac{e^x - e^2}{x - 2} = e^2$$

Tunjukkan limit tersebut dengan grafik, seperti contoh-contoh sebelumnya.

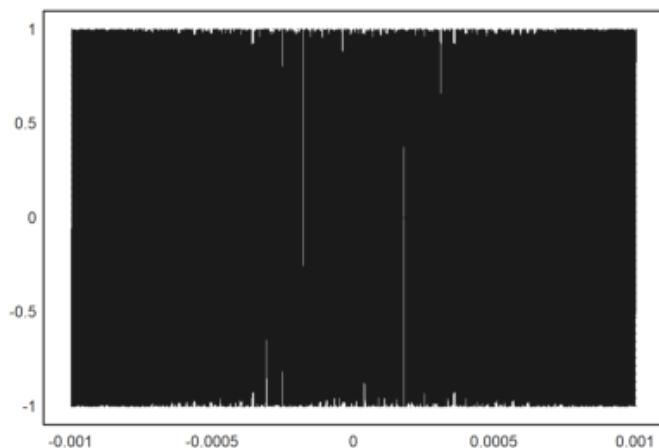
```
>$showev('limit(sin(1/x),x,0))
```

$$\lim_{x \rightarrow 0} \sin\left(\frac{1}{x}\right) = \text{ind}$$

```
>$showev('limit(sin(1/x),x,inf))
```

$$\lim_{x \rightarrow \infty} \sin\left(\frac{1}{x}\right) = 0$$

```
>plot2d("sin(1/x)",-0.001,0.001):
```



Latihan

Bukalah buku Kalkulus. Cari dan pilih beberapa (paling sedikit 5 fungsi berbeda tipe/bentuk/jenis) fungsi dari buku tersebut, kemudian definisikan di EMT pada baris-baris perintah berikut (jika perlu tambahkan lagi). Untuk setiap fungsi, hitung nilai limit fungsi tersebut di beberapa nilai dan di tak hingga. Gambar grafik fungsi tersebut untuk mengkonfirmasi nilai-nilai limit tersebut.

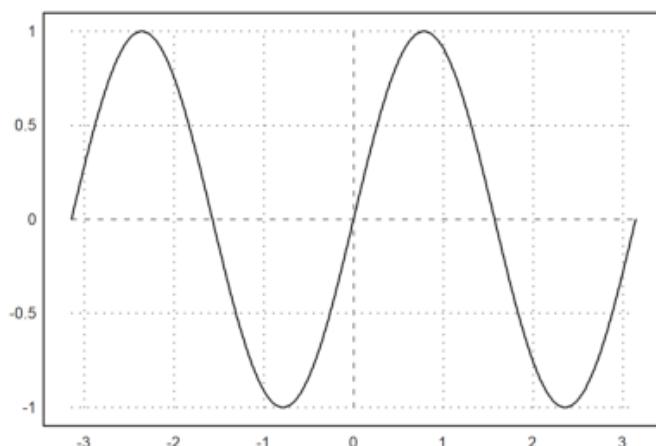
```
>$showev('limit(sin(2*x),x,0))
```

$$\lim_{x \rightarrow 0} \sin(2x) = 0$$

```
>$showev('limit(sin(2*x),x,pi/3))
```

$$\lim_{x \rightarrow \frac{\pi}{3}} \sin(2x) = \frac{\sqrt{3}}{2}$$

```
>plot2d("sin(2*x)",-pi,pi):
```

**2**

```
>$showev('limit(sin(x)^2,x,0))
```

$$\lim_{x \rightarrow 0} \sin^2 x = 0$$

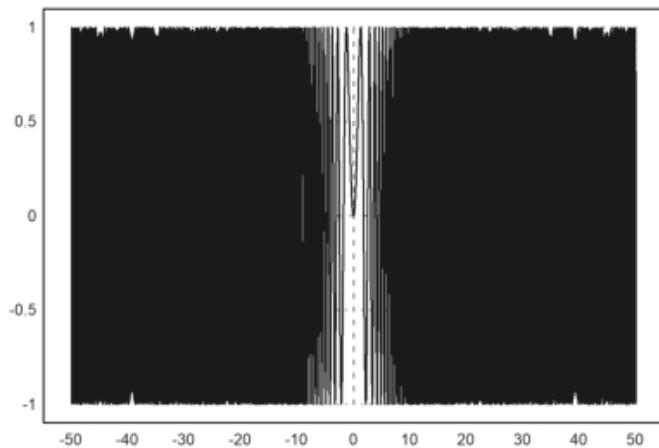
```
>$showev('limit(sin(x)^2,x,-2))
```

$$\lim_{x \rightarrow -2} \sin^2 x = \sin^2 2$$

```
>$showev('limit(sin(x)^2,x,-5))
```

$$\lim_{x \rightarrow -5} \sin^2 x = \sin^2 5$$

```
>plot2d("sin(x^2)",-50,50):
```



3

```
>$showev('limit(3*x^2+2*x+1,x,0))
```

$$\lim_{x \rightarrow 0} 3x^2 + 2x + 1 = 1$$

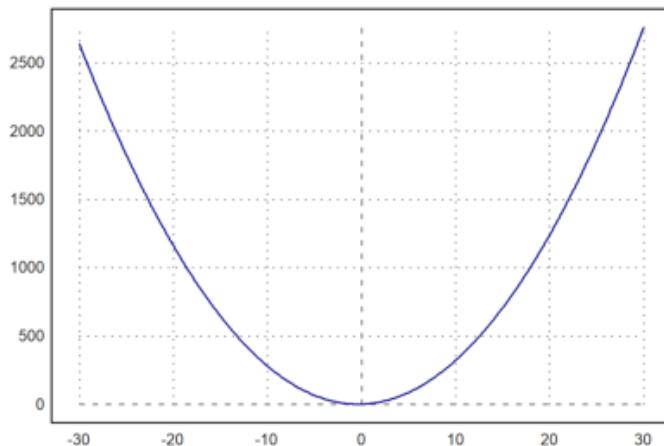
```
>$showev('limit(3*x^2+2*x+1,x,-5))
```

$$\lim_{x \rightarrow -5} 3x^2 + 2x + 1 = 66$$

```
>$showev('limit(3*x^2+2*x+1,x,-3))
```

$$\lim_{x \rightarrow -3} 3x^2 + 2x + 1 = 22$$

```
>plot2d("3*x^2+2*x+1",-30,30,color=blue):
```



4

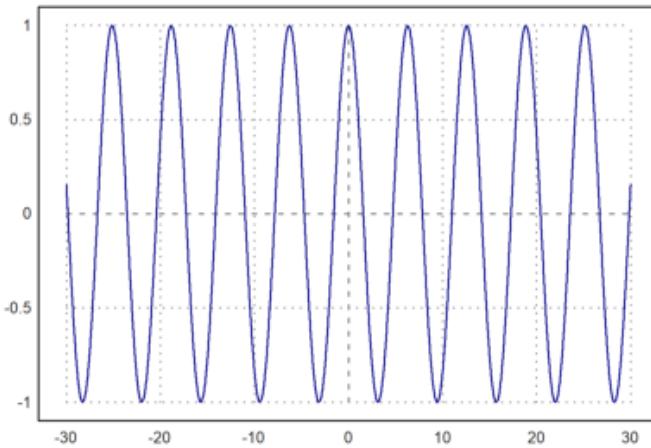
```
>$showev('limit(cos(x),x,0))
```

$$\lim_{x \rightarrow 0} \cos x = 1$$

```
>$showev('limit(cos(x),x,5))
```

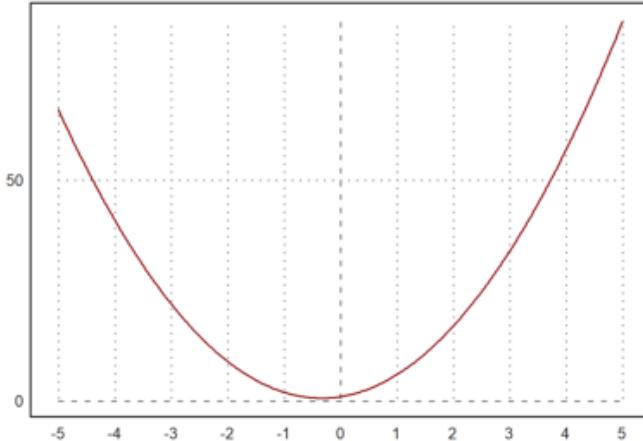
$$\lim_{x \rightarrow 5} \cos x = \cos 5$$

```
>plot2d("cos(x)",-30,30,color=blue):
```



5

```
>plot2d("3*x^2+2*x+1",-5,5,color=red):
```



Turunan Fungsi

Definisi turunan:

$$f'(x) = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}$$

Berikut adalah contoh-contoh menentukan turunan fungsi dengan menggunakan definisi turunan (limit).

```
>$showev('limit(((x+h)^2-x^2)/h,h,0)) // turunan x^2
```

$$\lim_{h \rightarrow 0} \frac{(x+h)^2 - x^2}{h} = 2x$$

```
>p &= expand((x+h)^2-x^2)|simplify; $p // pembilang dijabarkan dan disederhanakan
```

$$2hx + h^2$$

```
>q &=ratsimp(p/h); $q // ekspresi yang akan dihitung limitnya disederhanakan
```

$$2x + h$$

```
>$limit(q,h,0) // nilai limit sebagai turunan
```

$$2x$$

```
>$showev('limit(((x+h)^n-x^n)/h,h,0)) // turunan x^n
```

$$\lim_{h \rightarrow 0} \frac{(x+h)^n - x^n}{h} = nx^{n-1}$$

Mengapa hasilnya seperti itu? Tuliskan atau tunjukkan bahwa hasil limit tersebut benar, sehingga benar turunan fungsiya benar. Tulis penjelasan Anda di komentar ini. Sebagai petunjuk, ekspansikan $(x+h)^n$ dengan menggunakan teorema binomial.

```
>$showev('limit((sin(x+h)-sin(x))/h,h,0)) // turunan sin(x)
```

$$\lim_{h \rightarrow 0} \frac{\sin(x+h) - \sin x}{h} = \cos x$$

Mengapa hasilnya seperti itu? Tuliskan atau tunjukkan bahwa hasil limit tersebut benar, sehingga benar turunan fungsinya benar. Tulis penjelasan Anda di komentar ini. Sebagai petunjuk, ekspansikan $\sin(x+h)$ dengan menggunakan rumus jumlah dua sudut.

```
>$showev('limit((log(x+h)-log(x))/h,h,0)) // turunan log(x)
```

$$\lim_{h \rightarrow 0} \frac{\log(x+h) - \log x}{h} = \frac{1}{x}$$

Mengapa hasilnya seperti itu? Tuliskan atau tunjukkan bahwa hasil limit tersebut benar, sehingga benar turunan fungsinya benar. Tulis penjelasan Anda di komentar ini. Sebagai petunjuk, gunakan sifat-sifat logaritma dan hasil limit pada bagian sebelumnya di atas.

```
>$showev('limit((1/(x+h)-1/x)/h,h,0)) // turunan 1/x
```

$$\lim_{h \rightarrow 0} \frac{\frac{1}{x+h} - \frac{1}{x}}{h} = -\frac{1}{x^2}$$

```
>$showev('limit((E^(x+h)-E^x)/h,h,0)) // turunan f(x)=e^x
```

Answering "Is x an integer?" with "integer"
 Maxima is asking
 Acceptable answers are: yes, y, Y, no, n, N, unknown, uk
 Is x an integer?

Use assume!

Error in:

```
$showev('limit((E^(x+h)-E^x)/h,h,0)) // turunan f(x)=e^x ...  
^
```

Maxima bermasalah dengan limit:

$$\lim_{h \rightarrow 0} \frac{e^{x+h} - e^x}{h}.$$

Oleh karena itu diperlukan trik khusus agar hasilnya benar.

```
>$showev('limit((E^h-1)/h,h,0))
```

$$\lim_{h \rightarrow 0} \frac{e^h - 1}{h} = 1$$

```
>$showev('factor(E^(x+h)-E^x))
```

$$factor(e^{x+h} - e^x) = (e^h - 1) e^x$$

```
>$showev('limit(factor((E^(x+h)-E^x)/h),h,0)) // turunan f(x)=e^x
```

$$\left(\lim_{h \rightarrow 0} \frac{e^h - 1}{h} \right) e^x = e^x$$

```
>function f(x) &= x^x
```

$$\frac{x}{x}$$

```
>$showev('limit(f(x),x,0))
```

$$\lim_{x \rightarrow 0} x^x = 1$$

Silakan Anda gambar kurva

$$y = x^x.$$

```
>$showef('limit((f(x+h)-f(x))/h,h,0)) // merupakan turunan f(x)=x^x
```

$$\text{showef} \left(\lim_{h \rightarrow 0} \frac{(x+h)^{x+h} - x^x}{h} \right)$$

```
>$showev('limit((f(x+h)-f(x))/h,h,0)) // turunan f(x)=x^x
```

$$\lim_{h \rightarrow 0} \frac{(x+h)^{x+h} - x^x}{h} = \text{infinity}$$

Di sini Maxima juga bermasalah terkait limit:

$$\lim_{h \rightarrow 0} \frac{(x+h)^{x+h} - x^x}{h}.$$

Dalam hal ini diperlukan asumsi nilai x.

```
>&assume(x>0); $showev('limit((f(x+h)-f(x))/h,h,0)) // turunan f(x)=x^x
```

$$\lim_{h \rightarrow 0} \frac{(x+h)^{x+h} - x^x}{h} = x^x (\log x + 1)$$

Mengapa hasilnya seperti itu? Tuliskan atau tunjukkan bahwa hasil limit tersebut benar, sehingga benar turunan fungsinya benar. Tulis penjelasan Anda di komentar ini.

```
>&forget(x>0) // jangan lupa, lupakan asumsi untuk kembali ke semula
```

[x > 0]

```
>&forget(x<0)
```

[x < 0]

```
>&facts()
```

[]

```
>$showev('limit((asin(x+h)-asin(x))/h,h,0)) // turunan arcsin(x)
```

$$\lim_{h \rightarrow 0} \frac{\arcsin(x+h) - \arcsin x}{h} = \frac{1}{\sqrt{1-x^2}}$$

Mengapa hasilnya seperti itu? Tuliskan atau tunjukkan bahwa hasil limit tersebut benar, sehingga benar turunan fungsinya benar. Tulis penjelasan Anda di komentar ini.

```
>$showev('limit((tan(x+h)-tan(x))/h,h,0)) // turunan tan(x)
```

$$\lim_{h \rightarrow 0} \frac{\tan(x+h) - \tan x}{h} = \frac{1}{\cos^2 x}$$

Mengapa hasilnya seperti itu? Tuliskan atau tunjukkan bahwa hasil limit tersebut benar, sehingga benar turunan fungsinya benar. Tulis penjelasan Anda di komentar ini.

```
>function f(x) &= sinh(x) // definisikan f(x)=sinh(x)
```

$\sinh(x)$

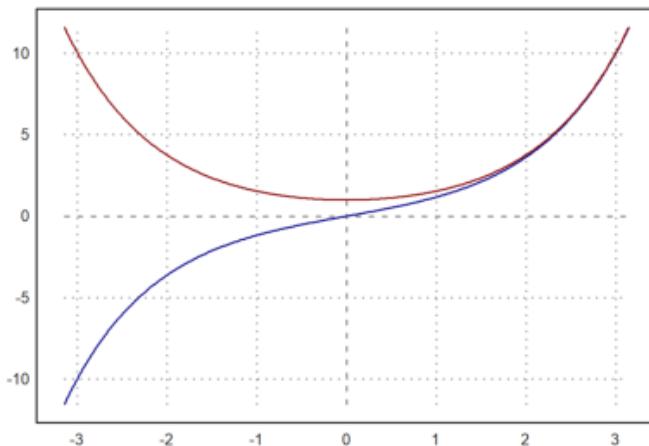
```
>function df(x) &= limit((f(x+h)-f(x))/h,h,0); $df(x) // df(x) = f'(x)
```

$$\frac{e^{-x} (e^{2x} + 1)}{2}$$

Hasilnya adalah $\cosh(x)$, karena

$$\frac{e^x + e^{-x}}{2} = \cosh(x).$$

```
>plot2d(["f(x)", "df(x)"], -pi, pi, color=[blue, red]):
```



```
>function f(x) &= sin(3*x^5+7)^2
```

$$\sin^2(3x^5 + 7)$$

```
>diff(f,3), diffc(f,3)
```

1198.32948904
1198.72863721

Apakah perbedaan diff dan diffc?

```
>$showev('diff(f(x),x))
```

$$\frac{d}{dx} \sin^2(3x^5 + 7) = 30x^4 \cos(3x^5 + 7) \sin(3x^5 + 7)$$

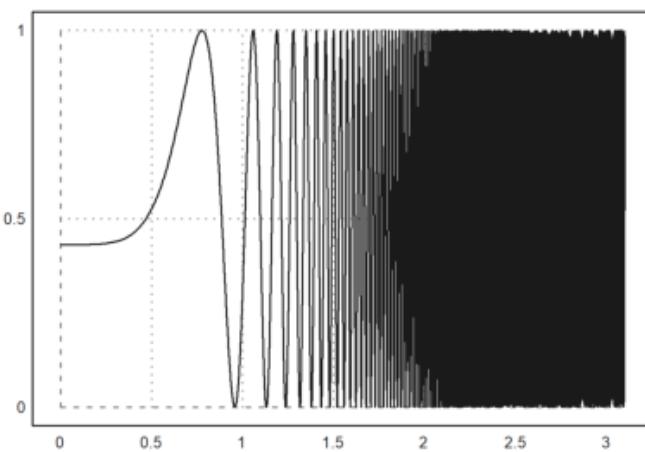
```
>$% with x=3
```

$$\%at \left(\frac{d}{dx} \sin^2(3x^5 + 7), x = 3 \right) = 2430 \cos 736 \sin 736$$

```
>${\tt float}(%)
```

$$\% \text{at} \left(\frac{d^{1.0}}{dx^{1.0}} \sin^2(3.0x^5 + 7.0), x = 3.0 \right) = 1198.728637211748$$

```
>plot2d(f, 0, 3.1);
```



```
>function f(x) &=5*cos(2*x)-2*x*sin(2*x) // mendefinisikan fungsi f
```

$$5 \cos(2x) - 2x \sin(2x)$$

```
>function df(x) &=diff(f(x),x) // fd(x) = f'(x)
```

$$- 12 \sin(2x) - 4x \cos(2x)$$

```
>${\tt \$}'f(1)=f(1), ${\tt \$float}(f(1)), ${\tt \$}'f(2)=f(2), ${\tt \$float}(f(2)) // nilai f(1) dan f(2)
```

$$f(1) = 5 \cos 2 - 2 \sin 2$$

$$-3.899329036387075$$

$$f(2) = 5 \cos 4 - 4 \sin 4$$

-0.2410081230863468

```
>xp=solve("df(x)",1,2,0) // solusi f'(x)=0 pada interval [1, 2]
```

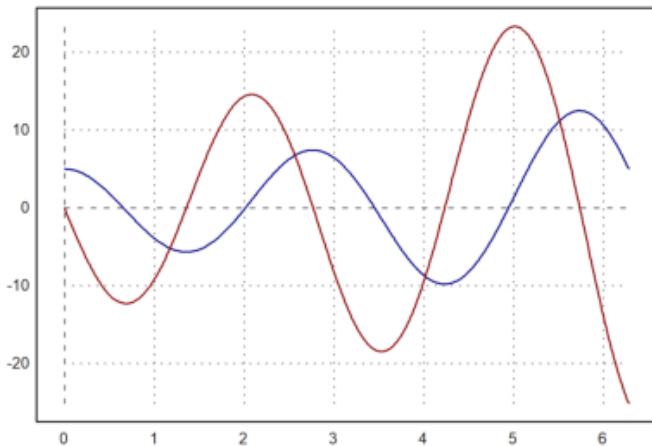
1.35822987384

```
>df(xp), f(xp) // cek bahwa f'(xp)=0 dan nilai ekstrim di titik tersebut
```

0

-5.67530133759

```
>plot2d(["f(x)", "df(x)", 0, 2*pi, color=[blue, red]): //grafik fungsi dan turu
```



Perhatikan titik-titik "puncak" grafik $y=f(x)$ dan nilai turunan pada saat grafik fungsinya mencapai titik "puncak" tersebut.

>

Latihan

Bukalah buku Kalkulus. Cari dan pilih beberapa (paling sedikit 5 fungsi berbeda tipe/bentuk/jenis) fungsi dari buku tersebut, kemudian definisikan di EMT pada baris-baris perintah berikut (jika perlu tambahkan lagi). Untuk setiap fungsi, tentukan turunannya dengan menggunakan definisi turunan (limit), menggunakan perintah diff, dan secara manual (langkah demi langkah yang dihitung dengan Maxima) seperti contoh-contoh di atas. Gambar grafik fungsi asli dan fungsi turunannya pada sumbu koordinat yang sama.

1

```
> function f(x) &= sin(x^2+6*x)^2
```

$$\sin^2(x^2 + 6x)$$

```
>$showev('limit(f(x), x, 0))
```

$$\lim_{x \rightarrow 0} \sin^2(x^2 + 6x) = 0$$

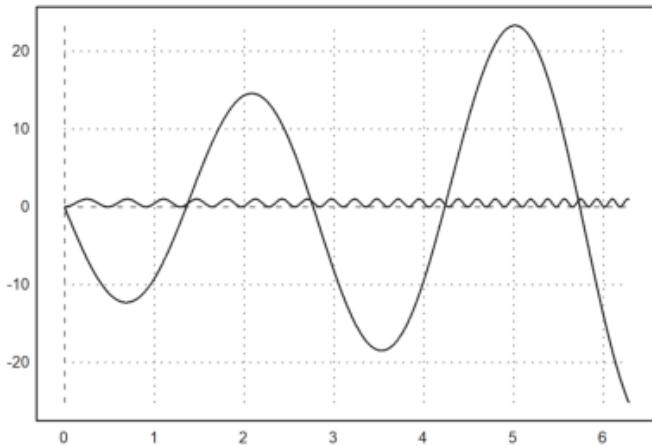
```
>$showev('limit(((x+h)^n - x^n)/h, h, 0))
```

$$\lim_{h \rightarrow 0} \frac{(x+h)^n - x^n}{h} = n x^{n-1}$$

```
>$df(x)
```

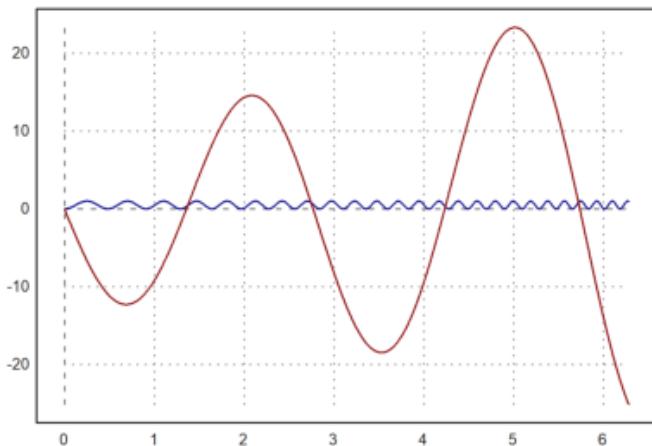
$$-12 \sin(2x) - 4x \cos(2x)$$

```
>plot2d(["f(x)", "df(x)", 0, 2*pi):
```



2

```
>plot2d(["f(x)", "df(x)"], 0, 2*pi, color=[blue, red]):
```



3

```
>function f(x) &= sin(x)^2
```

$$\sin^2(x)$$

```
>\$showev('limit(f(x),x,0))
```

$$\lim_{x \rightarrow 0} \sin^2 x = 0$$

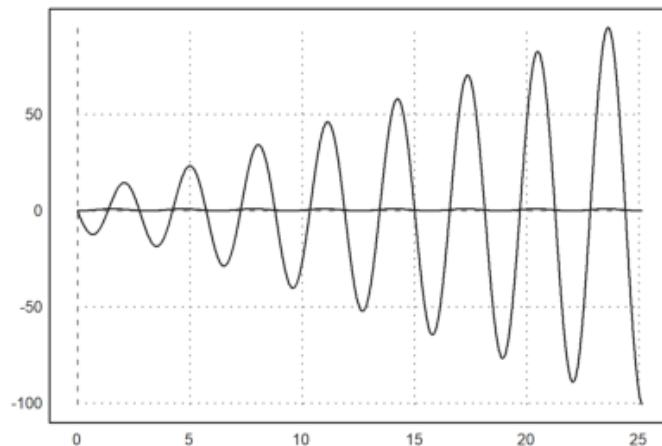
```
>\$showev('limit(f(x),x,-3))
```

$$\lim_{x \rightarrow -3} \sin^2 x = \sin^2 3$$

```
>\$df(x)
```

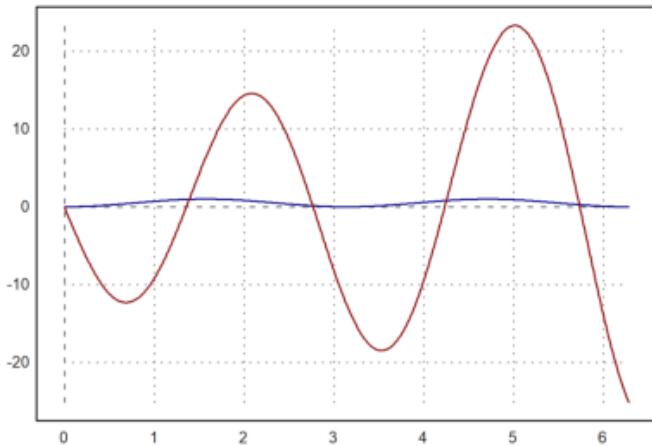
$$-12 \sin(2x) - 4x \cos(2x)$$

```
>plot2d(["f(x)", "df(x)"], 0, 8*pi):
```

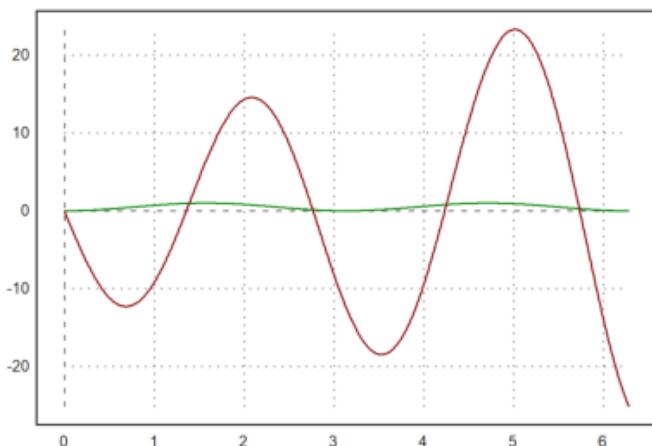


4

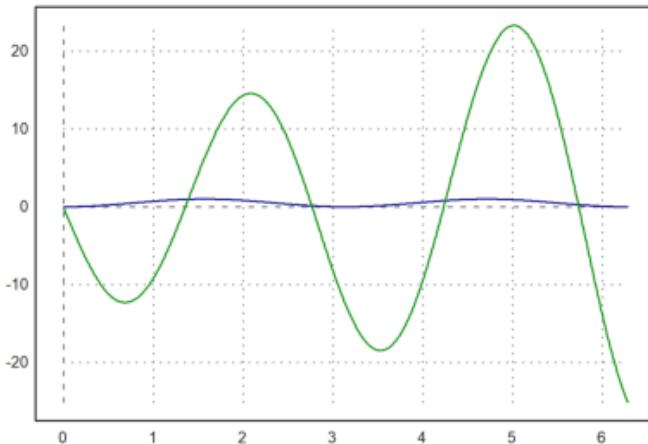
```
>plot2d(["f(x)", "df(x)"], 0, 2*pi, color=[blue, red]):
```

5

```
>plot2d(["f(x)", "df(x)", 0, 2*pi, color=[green, red]):
```

6

```
>plot2d(["f(x)", "df(x)", 0, 2*pi, color=[blue, green]):
```



Integral

EMT dapat digunakan untuk menghitung integral, baik integral tak tentu maupun integral tentu. Untuk integral tak tentu (simbolik) sudah tentu EMT menggunakan Maxima, sedangkan untuk perhitungan integral tentu EMT sudah menyediakan beberapa fungsi yang mengimplementasikan algoritma kuadratur (perhitungan integral tentu menggunakan metode numerik).

Pada notebook ini akan ditunjukkan perhitungan integral tentu dengan menggunakan Teorema Dasar Kalkulus:

$$\int_a^b f(x) dx = F(b) - F(a), \quad \text{dengan } F'(x) = f(x).$$

Fungsi untuk menentukan integral adalah `integrate`. Fungsi ini dapat digunakan untuk menentukan, baik integral tentu maupun tak tentu (jika fungsinya memiliki antiderivatif). Untuk perhitungan integral tentu fungsi `integrate` menggunakan metode numerik (kecuali fungsinya tidak integrabel, kita tidak akan menggunakan metode ini).

```
>$showev('integrate(x^n, x))
```

Answering "Is n equal to -1?" with "no"

$$\int x^n dx = \frac{x^{n+1}}{n+1}$$

```
>$showev('integrate(1/(1+x), x))
```

$$\int \frac{1}{x+1} dx = \log(x+1)$$

```
>$showev('integrate(1/(1+x^2),x))
```

$$\int \frac{1}{x^2 + 1} dx = \arctan x$$

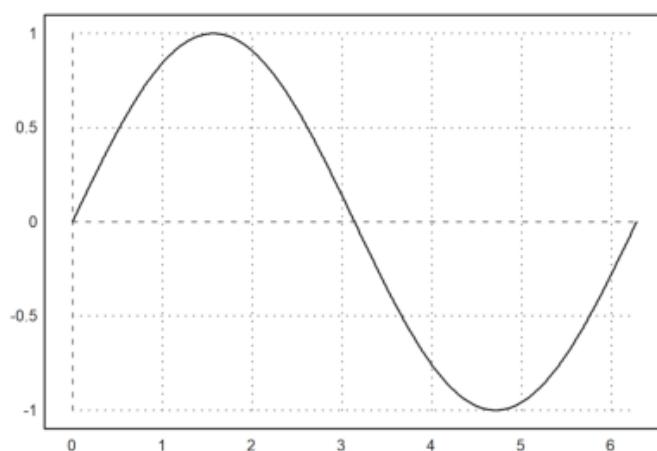
```
>$showev('integrate(1/sqrt(1-x^2),x))
```

$$\int \frac{1}{\sqrt{1 - x^2}} dx = \arcsin x$$

```
>$showev('integrate(sin(x),x,0,pi))
```

$$\int_0^\pi \sin x dx = 2$$

```
>plot2d("sin(x)",0,2*pi):
```



```
>$showev('integrate(sin(x),x,a,b))
```

$$\int_a^b \sin x dx = \cos a - \cos b$$

```
>$showev('integrate(x^n,x,a,b))
```

Answering "Is n positive, negative or zero?" with "positive"

$$\int_a^b x^n dx = \frac{b^{n+1}}{n+1} - \frac{a^{n+1}}{n+1}$$

```
>$showev('integrate(x^2*sqrt(2*x+1),x))
```

$$\int x^2 \sqrt{2x+1} dx = \frac{(2x+1)^{\frac{7}{2}}}{28} - \frac{(2x+1)^{\frac{5}{2}}}{10} + \frac{(2x+1)^{\frac{3}{2}}}{12}$$

```
>$showev('integrate(x^2*sqrt(2*x+1),x,0,2))
```

$$\int_0^2 x^2 \sqrt{2x+1} dx = \frac{25^{\frac{5}{2}}}{21} - \frac{2}{105}$$

```
>$ratsimp(%)
```

$$\int_0^2 x^2 \sqrt{2x+1} dx = \frac{25^{\frac{7}{2}} - 2}{105}$$

```
>$showev('integrate((sin(sqrt(x)+a)*E^sqrt(x))/sqrt(x),x,0,pi^2))
```

$$\int_0^{\pi^2} \frac{\sin(\sqrt{x} + a) e^{\sqrt{x}}}{\sqrt{x}} dx = (-e^\pi - 1) \sin a + (e^\pi + 1) \cos a$$

```
>$factor(%)
```

$$\int_0^{\pi^2} \frac{\sin(\sqrt{x} + a) e^{\sqrt{x}}}{\sqrt{x}} dx = (-e^\pi - 1) (\sin a - \cos a)$$

```
>function map f(x) &= E^(-x^2)
```

$$\frac{E^{-x^2}}{2}$$

```
>$showev('integrate(f(x),x))
```

$$\int e^{-x^2} dx = \frac{\sqrt{\pi} \operatorname{erf}(x)}{2}$$

Fungsi f tidak memiliki antiturunan, integralnya masih memuat integral lain.

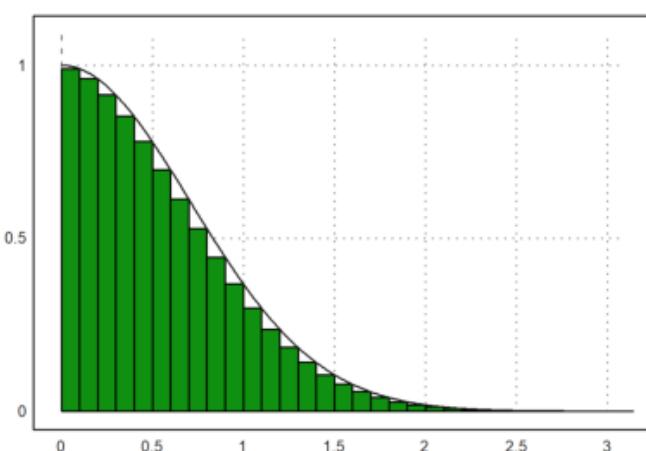
$$\operatorname{erf}(x) = \int \frac{e^{-x^2}}{\sqrt{\pi}} dx.$$

Kita tidak dapat menggunakan teorema Dasar kalkulus untuk menghitung integral tentu fungsi tersebut jika semua batasnya berhingga. Dalam hal ini dapat digunakan metode numerik (rumus kuadratur).

Misalkan kita akan menghitung:

$$\int_0^\pi e^{-x^2} dx$$

```
>x=0:0.1:pi-0.1; plot2d(x,f(x+0.1),>bar); plot2d("f(x)",0,pi,>add):
```



$$\int_0^{\pi} e^{-x^2} dx$$

dapat dihampiri dengan jumlah luas persegi-persegi panjang di bawah kurva $y=f(x)$ tersebut. Langkah-langkahnya adalah sebagai berikut.

```
>t &= makelist(a,a,0,pi-0.1,0.1); // t sebagai list untuk menyimpan nilai-nilai f(x) untuk x = 0.1, 0.2, 0.3, ..., 3.2.
>fx &= makelist(f(t[i]+0.1),i,1,length(t)); // simpan nilai-nilai f(x)
>/> jangan menggunakan x sebagai list, kecuali Anda pakar Maxima!
```

Hasilnya adalah:

maxima: 'integrate(f(x),x,0,pi) = 0.1*sum(fx[i],i,1,length(fx))

Jumlah tersebut diperoleh dari hasil kali lebar sub-subinterval (=0.1) dan jumlah nilai-nilai $f(x)$ untuk $x = 0.1, 0.2, 0.3, \dots, 3.2$.

```
>0.1*sum(f(x+0.1)) // cek langsung dengan perhitungan numerik EMT
```

0.836219610253

Untuk mendapatkan nilai integral tentu yang mendekati nilai sebenarnya, lebar sub-intervalnya dapat diperkecil lagi, sehingga daerah di bawah kurva tertutup semuanya, misalnya dapat digunakan lebar subinterval 0.001. (Silakan dicoba!)

Meskipun Maxima tidak dapat menghitung integral tentu fungsi tersebut untuk batas-batas yang berhingga, namun integral tersebut dapat dihitung secara eksak jika batas-batasnya tak hingga. Ini adalah salah satu keajaiban di dalam matematika, yang terbatas tidak dapat dihitung secara eksak, namun yang tak hingga malah dapat dihitung secara eksak.

```
>$showev('integrate(f(x),x,0,inf))
```

$$\int_0^{\infty} e^{-x^2} dx = \frac{\sqrt{\pi}}{2}$$

Tunjukkan kebenaran hasil di atas!

Berikut adalah contoh lain fungsi yang tidak memiliki antiderivatif, sehingga integral tentunya hanya dapat dihitung dengan metode numerik.

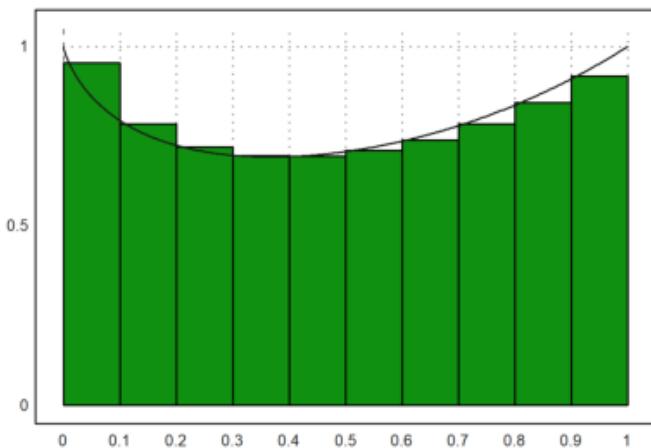
```
>function f(x) &= x^x
```

x
x

```
>$showev('integrate(f(x),x,0,1))
```

$$\int_0^1 x^x \, dx = \int_0^1 x^x \, dx$$

```
>x=0:0.1:1-0.01; plot2d(x,f(x+0.01),>bar); plot2d("f(x)",0,1,>add):
```



Maxima gagal menghitung integral tentu tersebut secara langsung menggunakan perintah integrate. Berikut kita lakukan seperti contoh sebelumnya untuk mendapat hasil atau pendekatan nilai integral tentu tersebut.

```
>t &= makelist(a,a,0,1-0.01,0.01);
>fx &= makelist(f(t[i]+0.01),i,1,length(t));
```

maxima: 'integrate(f(x),x,0,1) = 0.01*sum(fx[i],i,1,length(fx))
Apakah hasil tersebut cukup baik? perhatikan gambarnya.

```
>function f(x) &= sin(3*x^5+7)^2
```

$$\sin^2(3x^5 + 7)$$

```
>integrate(f,0,1)
```

0.542581176074

```
>&showev('integrate(f(x),x,0,1))
```

$$\begin{aligned} & \frac{1}{\Gamma(2)} \int_0^{3x^5 + 7} \sin^5 u \, du = \frac{\sin(14) \sin(\pi/10)}{10^6} \\ & - \left(\frac{6 \operatorname{gamma_incomplete}(-, 6I) + 6 \operatorname{gamma_incomplete}(-, -6I)}{5} \right. \\ & \quad \left. \sin(14) + \frac{(6 - I) \operatorname{gamma_incomplete}(-, 6I)}{5} \right. \\ & \quad \left. - \frac{6 - I \operatorname{gamma_incomplete}(-, -6I) \cos(14)}{5} \sin(\pi/10) - 60/120 \right) \end{aligned}$$

```
>&float(%)
```

$$\begin{aligned} & 1.0 \\ & / \\ & [\int_0^{3.0x^5 + 7.0} \sin(u) \, du = \\ &] \\ & / \\ & 0.0 \\ & 0.09820784258795788 - 0.00833333333333333 \\ & (0.3090169943749474 (0.1367372182078336 \end{aligned}$$

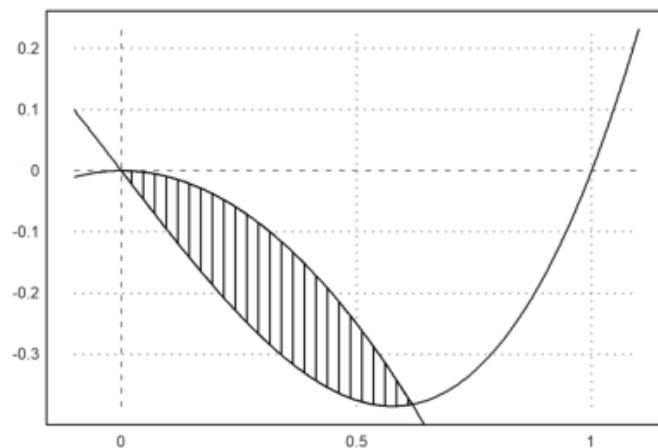
```
(4.192962712629476 I gamma_incomplete(0.2, 6.0 I)
 - 4.192962712629476 I gamma_incomplete(0.2, - 6.0 I))
 + 0.9906073556948704 (4.192962712629476 gamma_incomplete(0.2, 6.0 I)
 + 4.192962712629476 gamma_incomplete(0.2, - 6.0 I))) - 60.0)
```

```
>$showev('integrate(x*exp(-x),x,0,1)) // Integral tentu (eksak)
```

$$\int_0^1 x e^{-x} dx = 1 - 2e^{-1}$$

Aplikasi Integral Tentu

```
>plot2d("x^3-x",-0.1,1.1); plot2d("-x^2",>add); ...
>b=solve("x^3-x+x^2",0.5); x=linspace(0,b,200); xi=flipx(x); ...
>plot2d(x|xi,x^3-x|-xi^2,>filled,style="|",fillcolor=1,>add); // Plot daerah
```



```
>a=solve("x^3-x+x^2",0), b=solve("x^3-x+x^2",1) // absis titik-titik potong
```

```
0
0.61803398875
```

```
>integrate("(-x^2)-(x^3-x)",a,b) // luas daerah yang diarsir
```

```
0.0758191713542
```

Hasil tersebut akan kita bandingkan dengan perhitungan secara analitik.

```
>a &= solve( (-x^2) - (x^3-x) , x); $a // menentukan absis titik potong kedua ku
```

$$\left[x = \frac{-\sqrt{5} - 1}{2}, x = \frac{\sqrt{5} - 1}{2}, x = 0 \right]$$

```
>$showev('integrate(-x^2-x^3+x, x, 0, (sqrt(5)-1)/2)) // Nilai integral secara
```

$$\int_0^{\frac{\sqrt{5}-1}{2}} -x^3 - x^2 + x \, dx = \frac{13 - 5^{\frac{3}{2}}}{24}$$

```
>$float(%)
```

$$\int_{0.0}^{0.6180339887498949} -1.0 x^3 - 1.0 x^2 + x \, dx = 0.07581917135421037$$

Panjang Kurva

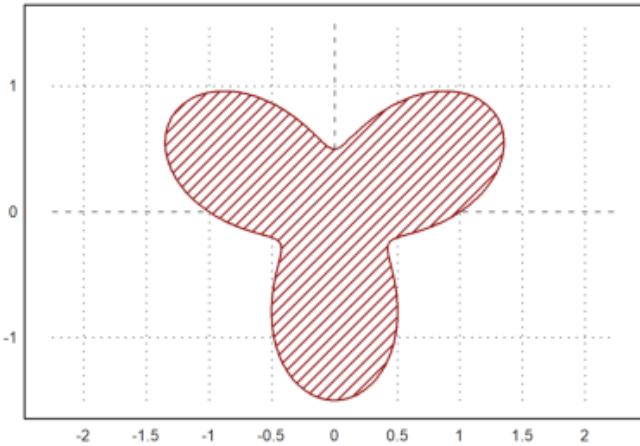
Hitunglah panjang kurva berikut ini dan luas daerah di dalam kurva tersebut.

$$\gamma(t) = (r(t) \cos(t), r(t) \sin(t))$$

dengan

$$r(t) = 1 + \frac{\sin(3t)}{2}, \quad 0 \leq t \leq 2\pi.$$

```
>t=linspace(0,2pi,1000); r=1+sin(3*t)/2; x=r*cos(t); y=r*sin(t); ...
>plot2d(x,y,>filled,fillcolor=red,style="/" ,r=1.5): // Kita gambar kurvanya
```



```
>function r(t) &= 1+sin(3*t)/2; $'r(t)=r(t)
```

$r([0, 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1, 0.11, 0.12, 0.13, 0.14, 0.15, 0.16, 0.17, 0.18, 0.19, 0.2, 0.21])$

```
>function fx(t) &= r(t)*cos(t); $'fx(t)=fx(t)
```

$fx([0, 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1, 0.11, 0.12, 0.13, 0.14, 0.15, 0.16, 0.17, 0.18, 0.19, 0.2, 0.21])$

```
>function fy(t) &= r(t)*sin(t); $'fy(t)=fy(t)
```

$fy([0, 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1, 0.11, 0.12, 0.13, 0.14, 0.15, 0.16, 0.17, 0.18, 0.19, 0.2, 0.21])$

```
>function ds(t) &= trigreduce(radcan(sqrt(diff(fx(t),t)^2+diff(fy(t),t)^2)))
```

Maxima said:

```
diff: second argument must be a variable; found errexp1
-- an error. To debug this try: debugmode(true);
```

Error in:

```
... e(radcan(sqrt(diff(fx(t),t)^2+diff(fy(t),t)^2))); $'ds(t)=ds(t ...  
^
```

```
>$integrate(ds(x),x,0,2*pi) //panjang (keliling) kurva
```

$$\int_0^{2\pi} ds(x) \, dx$$

Maxima gagal melakukan perhitungan eksak integral tersebut.

Berikut kita hitung integralnya secara numerik dengan perintah EMT.

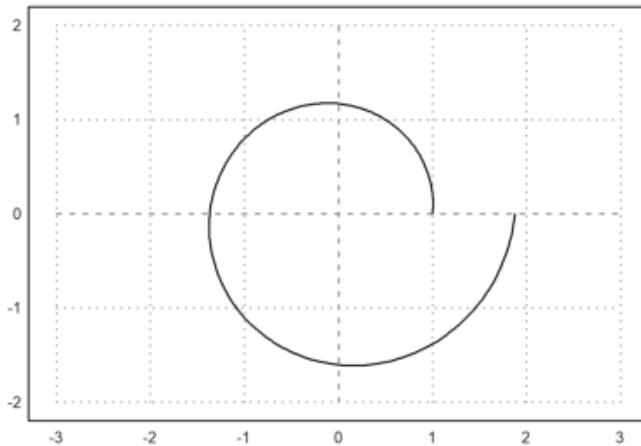
```
>integrate("ds(x)",0,2*pi)
```

```
Function ds not found.
Try list ... to find functions!
Error in expression: ds(x)
%mapexpression:
    return expr(x,args());
Error in map.
%evalexpression:
    if maps then return %mapexpression1(x,f$;args());
gauss:
    if maps then y=%evalexpression(f$,a+h-(h*xn)',maps;args());
adaptivegauss:
    t1=gauss(f$,c,c+h;args(),=maps);
Try "trace errors" to inspect local variables after errors.
integrate:
    return adaptivegauss(f$,a,b,eps*1000;args(),=maps);
```

Spiral Logaritmik

$$x = e^{ax} \cos x, \quad y = e^{ax} \sin x.$$

```
>a=0.1; plot2d("exp(a*x)*cos(x)", "exp(a*x)*sin(x)", r=2, xmin=0, xmax=2*pi):
```



```
>&kill(a) // hapus expresi a
```

done

```
>function fx(t) &= exp(a*t)*cos(t); $'fx(t)=fx(t)
```

$fx([0, 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1, 0.11, 0.12, 0.13, 0.14, 0.15, 0.16, 0.17, 0.18, 0.19, 0.2,$

```
>function fy(t) &= exp(a*t)*sin(t); $'fy(t)=fy(t)
```

$fy([0, 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1, 0.11, 0.12, 0.13, 0.14, 0.15, 0.16, 0.17, 0.18, 0.19, 0.2,$

```
>function df(t) &= trigreduce(radcan(sqrt(diff(fx(t),t)^2+diff(fy(t),t)^2)))
```

Maxima said:

```
diff: second argument must be a variable; found errexp1
-- an error. To debug this try: debugmode(true);
```

Error in:

```
... e(radcan(sqrt(diff(fx(t),t)^2+diff(fy(t),t)^2))); $'df(t)=df(t ...  
^
```

```
>S &=integrate(df(t),t,0,2*pi); $S // panjang kurva (spiral)
```

Maxima said:

```
defint: variable of integration cannot be a constant; found errexp1
-- an error. To debug this try: debugmode(true);
```

Error in:

```
S &=integrate(df(t),t,0,2*pi); $S // panjang kurva (spiral) ...  
^
```

```
>S(a=0.1) // Panjang kurva untuk a=0.1
```

Function S not found.

Try list ... to find functions!

Error in:

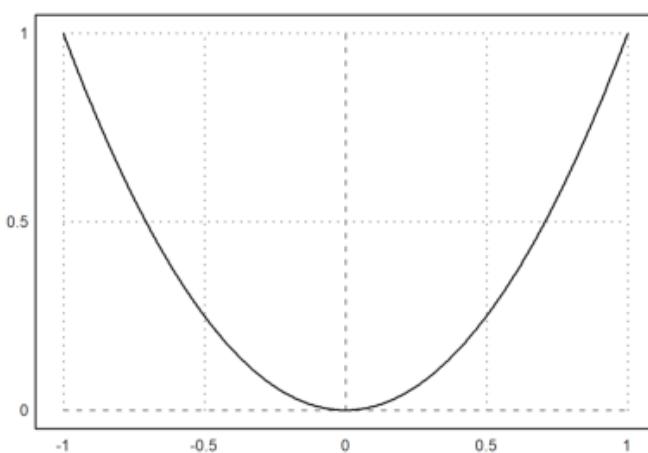
```
S(a=0.1) // Panjang kurva untuk a=0.1 ...  
^
```

Soal:

Tunjukkan bahwa keliling lingkaran dengan jari-jari r adalah $K=2\pi r$.

Berikut adalah contoh menghitung panjang parabola.

```
>plot2d("x^2", xmin=-1, xmax=1):
```



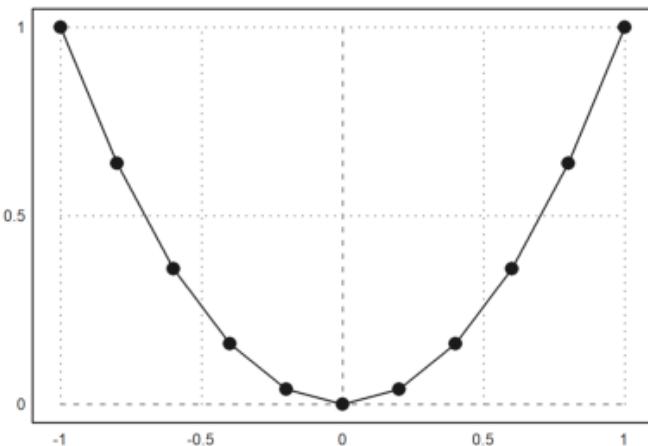
```
>$showev('integrate(sqrt(1+diff(x^2,x)^2),x,-1,1))
```

$$\int_{-1}^1 \sqrt{4x^2 + 1} dx = \frac{\operatorname{asinh} 2 + 2\sqrt{5}}{2}$$

```
> $float (%)
```

$$\int_{-1.0}^{1.0} \sqrt{4.0x^2 + 1.0} dx = 2.957885715089195$$

```
>x=-1:0.2:1; y=x^2; plot2d(x,y); ...
> plot2d(x,y,points=1,style="o#",add=1):
```



Panjang tersebut dapat dihampiri dengan menggunakan jumlah panjang ruas-ruas garis yang menghubungkan titik-titik pada parabola tersebut.

```
>i=1:cols(x)-1; sum(sqrt((x[i+1]-x[i])^2+(y[i+1]-y[i])^2))
```

2.95191957027

Hasilnya mendekati panjang yang dihitung secara eksak. Untuk mendapatkan hampiran yang cukup akurat, jarak antar titik dapat diperkecil, misalnya 0.1, 0.05, 0.01, dan seterusnya. Cobalah Anda ulangi perhitungannya dengan nilai-nilai tersebut.

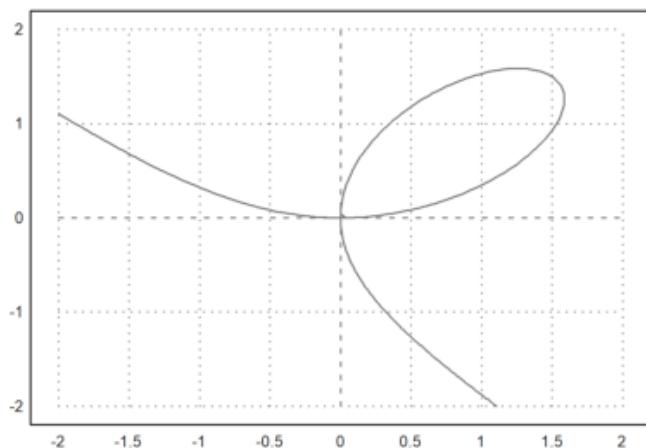
Berikut diberikan contoh perhitungan panjang kurva menggunakan koordinat Kartesius. Kita akan hitung panjang kurva dengan persamaan implisit:

$$x^3 + y^3 - 3xy = 0.$$

```
>z &= x^3+y^3-3*x*y; $z
```

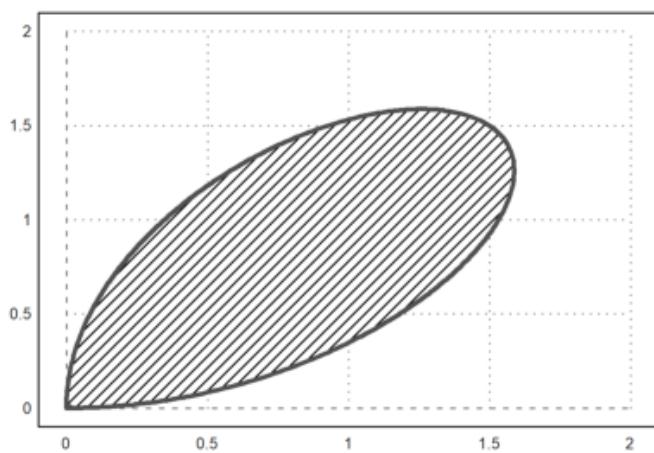
$$y^3 - 3xy + x^3$$

```
>plot2d(z, r=2, level=0, n=100) :
```



Kita tertarik pada kurva di kuadran pertama.

```
>plot2d(z, a=0, b=2, c=0, d=2, level=[-10;0], n=100, contourwidth=3, style="/") :
```



Kita selesaikan persamaannya untuk x.

```
>$z with y=l*x, sol &= solve(%,x); $sol
```

$$l^3 x^3 + x^3 - 3 l x^2$$

$$\left[x = \frac{3l}{l^3 + 1}, x = 0 \right]$$

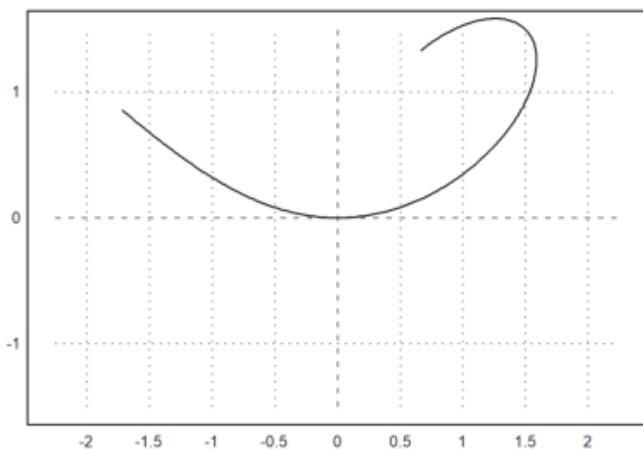
Kita gunakan solusi tersebut untuk mendefinisikan fungsi dengan Maxima.

```
>function f(l) &= rhs(sol[1]); $' f(l)=f(l)
```

$$f(l) = \frac{3l}{l^3 + 1}$$

Fungsi tersebut juga dapat digunakan untuk menggambar kurvanya. Ingat, bahwa fungsi tersebut adalah nilai x dan nilai y=l*x, yakni x=f(l) dan y=l*f(l).

```
>plot2d(&f(x),&x*f(x),xmin=-0.5,xmax=2,a=0,b=2,c=0,d=2,r=1.5):
```



Elemen panjang kurva adalah:

$$ds = \sqrt{f'(l)^2 + (l f'(l) + f(l))^2}.$$

```
>function ds(l) &= ratsimp(sqrt(diff(f(l),l)^2+diff(l*f(l),l)^2)); $'ds(l)=
```

$$ds(l) = \frac{\sqrt{9l^8 + 36l^6 - 36l^5 - 36l^3 + 36l^2 + 9}}{\sqrt{l^{12} + 4l^9 + 6l^6 + 4l^3 + 1}}$$

```
>$integrate(ds(l),l,0,1)
```

$$\int_0^1 \frac{\sqrt{9l^8 + 36l^6 - 36l^5 - 36l^3 + 36l^2 + 9}}{\sqrt{l^{12} + 4l^9 + 6l^6 + 4l^3 + 1}} dl$$

Integral tersebut tidak dapat dihitung secara eksak menggunakan Maxima. Kita hitung integral tersebut secara numerik dengan Euler. Karena kurva simetris, kita hitung untuk nilai variabel integrasi dari 0 sampai 1, kemudian hasilnya dikalikan 2.

```
>2*integrate("ds(x)",0,1)
```

4.91748872168

```
>2*romberg(&ds(x),0,1) // perintah Euler lain untuk menghitung nilai hampiran
```

4.91748872168

Perhitungan di atas dapat dilakukan untuk sebarang fungsi x dan y dengan mendefinisikan fungsi EMT, misalnya kita beri nama panjangkurva. Fungsi ini selalu memanggil Maxima untuk menurunkan fungsi yang diberikan.

```
>function panjangkurva(fx,fy,a,b) ...
```

```
ds=mxm("sqrt(diff(@fx,x)^2+diff(@fy,x)^2)");
return romberg(ds,a,b);
endfunction
```

```
>panjangkurva("x","x^2",-1,1) // cek untuk menghitung panjang kurva parabol
```

2.95788571509

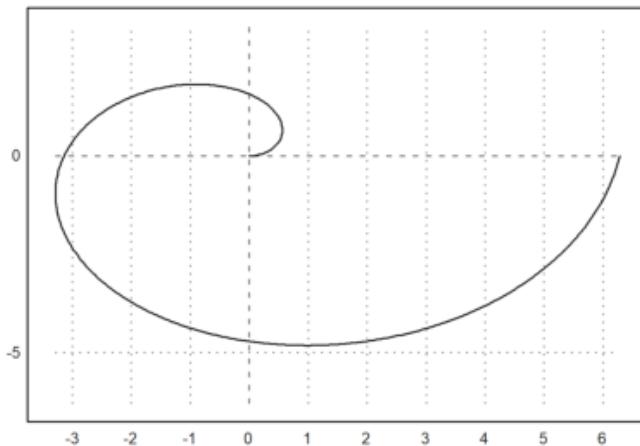
Bandingkan dengan nilai eksak di atas.

```
>2*panjangkurva(mxm("f(x)",mxm("x*f(x)",0,1)) // cek contoh terakhir, band
```

4.91748872168

Kita hitung panjang spiral Archimedes berikut ini dengan fungsi tersebut.

```
>plot2d("x*cos(x)", "x*sin(x)", xmin=0, xmax=2*pi, square=1):
```



```
>panjangkurva("x*cos(x)", "x*sin(x)", 0, 2*pi)
```

21.2562941482

Berikut kita definisikan fungsi yang sama namun dengan Maxima, untuk perhitungan eksak.

```
>&kill(ds,x,fx,fy)
```

done

```
>function ds(fx,fy) &=& sqrt(diff(fx,x)^2+diff(fy,x)^2)
```

$$\sqrt{(\cos x - x \sin x)^2 + (\sin x + x \cos x)^2}$$

```
>sol &= ds(x*cos(x),x*sin(x)); $sol // Kita gunakan untuk menghitung panjan
```

$$\sqrt{(\cos x - x \sin x)^2 + (\sin x + x \cos x)^2}$$

```
>$sol | trigreduce | expand, $integrate(% ,x,0,2*pi), %()
```

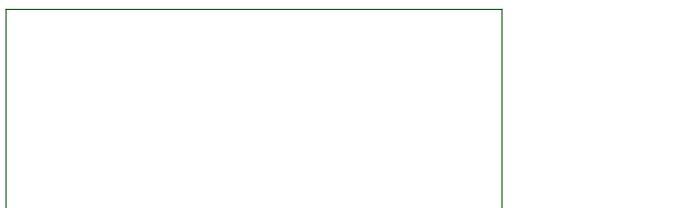
$$\sqrt{x^2 + 1}$$

$$\frac{\operatorname{asinh}(2\pi) + 2\pi\sqrt{4\pi^2 + 1}}{2}$$

21.2562941482

Hasilnya sama dengan perhitungan menggunakan fungsi EMT.
Berikut adalah contoh lain penggunaan fungsi Maxima tersebut.

```
>plot2d("3*x^2-1", "3*x^3-1", xmin=-1/sqrt(3), xmax=1/sqrt(3), square=1);
```



images/EMT4Kalkulus_Saphira-163.png

```
>sol &= radcan(ds(3*x^2-1, 3*x^3-1)); $sol
```

$$3x\sqrt{9x^2 + 4}$$

```
>$showev('integrate(sol,x,0,1/sqrt(3))), $2*float(%); // panjang kurva di at
```

$$3 \int_0^{\frac{1}{\sqrt{3}}} x \sqrt{9x^2 + 4} dx = 3 \left(\frac{7}{27} - \frac{8}{27} \right)$$

$$6.0 \int_{0.0}^{0.5773502691896258} x \sqrt{9.0x^2 + 4.0} dx = 2.337835372767141$$

Sikloid

Berikut kita akan menghitung panjang kurva lintasan (sikloid) suatu titik pada lingkaran yang berputar ke kanan pada permukaan datar. Misalkan jari-jari lingkaran tersebut adalah r . Posisi titik pusat lingkaran pada saat t adalah:

$$(rt, r).$$

Misalkan posisi titik pada lingkaran tersebut mula-mula $(0,0)$ dan posisinya pada saat t adalah:

$$(r(t - \sin(t)), r(1 - \cos(t))).$$

Berikut kita plot lintasan tersebut dan beberapa posisi lingkaran ketika $t=0, t=\pi/2, t=r*\pi$.

```
>x &= r*(t-sin(t))
```

```
[0, 1.66665833335744e-7 r, 1.33330666692022e-6 r,
4.499797504338432e-6 r, 1.066581336583994e-5 r,
2.083072932167196e-5 r, 3.599352055540239e-5 r,
5.71526624672386e-5 r, 8.530603082730626e-5 r,
1.214508019889565e-4 r, 1.665833531718508e-4 r,
2.216991628251896e-4 r, 2.877927110806339e-4 r,
3.658573803051457e-4 r, 4.568853557635201e-4 r,
5.618675264007778e-4 r, 6.817933857540259e-4 r,
8.176509330039827e-4 r, 9.704265741758145e-4 r,
0.001141105023499428 r, 0.001330669204938795 r,
0.001540100153900437 r, 0.001770376919130678 r,
0.002022476464811601 r, 0.002297373572865413 r,
0.002596040745477063 r, 0.002919448107844891 r,
0.00326856331168871 r, 0.003644351435886262 r,
```

0.004047774895164447 r, 0.004479793338660443 r, 0.004941363565565 r,
 0.005433439383882244 r, 0.005956971605131645 r,
 0.006512907859185624 r, 0.007102192544548636 r,
 0.007725766724910044 r, 0.00838456803503801 r,
 0.009079530587017326 r, 0.009811584876838586 r, 0.0105816576913495 r,
 0.01139067201557714 r, 0.01223954694042984 r, 0.01312919757078923 r,
 0.01406053493400045 r, 0.01503446588876983 r, 0.01605189303448024 r,
 0.01711371462093175 r, 0.01822082445851714 r, 0.01937411182884202 r,
 0.02057446139579705 r, 0.02182275311709253 r, 0.02311986215626333 r,
 0.02446665879515308 r, 0.02586400834688696 r, 0.02731277106934082 r,
 0.02881380207911666 r, 0.03036795126603076 r, 0.03197606320812652 r,
 0.0336389770872163 r, 0.03535752660496472 r, 0.03713253989951881 r,
 0.03896483946269502 r, 0.0408552420577305 r, 0.04280455863760801 r,
 0.04481359426396048 r, 0.04688314802656623 r, 0.04901401296344043 r,
 0.05120697598153157 r, 0.05346281777803219 r, 0.05578231276230905 r,
 0.05816622897846346 r, 0.06061532802852698 r, 0.0631303649963022 r,
 0.06571208837185505 r, 0.06836123997666599 r, 0.07107855488944881 r,
 0.07386476137264342 r, 0.07672058079958999 r, 0.07964672758239233 r,
 0.08264390910047736 r, 0.0857128256298576 r, 0.08885417027310427 r,
 0.09206862889003742 r, 0.09535688002914089 r, 0.0987195948597075 r,
 0.1021574371047232 r, 0.1056710629744951 r, 0.1092611211010309 r,
 0.1129282524731764 r, 0.1166730903725168 r, 0.1204962603100498 r,
 0.1243983799636342 r, 0.1283800591162231 r, 0.1324418995948859 r,
 0.1365844952106265 r, 0.140808431699002 r, 0.1451142866615502 r,
 0.1495026295080298 r, 0.1539740213994798 r]

```
>y &= r*(1-cos(t))
```

[0, 4.999958333473664e-5 r, 1.999933334222437e-4 r,
 4.499662510124569e-4 r, 7.998933390220841e-4 r,
 0.001249739605033717 r, 0.00179946006479581 r,
 0.002448999746720415 r, 0.003198293697380561 r,
 0.004047266988005727 r, 0.004995834721974179 r,
 0.006043902043303184 r, 0.00719136414613375 r, 0.00843810628521191 r,
 0.009784003787362772 r, 0.01122892206395776 r, 0.01277271662437307 r,
 0.01441523309043924 r, 0.01615630721187855 r, 0.01799576488272969 r,
 0.01993342215875837 r, 0.02196908527585173 r, 0.02410255066939448 r,
 0.02633360499462523 r, 0.02866202514797045 r, 0.03108757828935527 r,
 0.03361002186548678 r, 0.03622910363410947 r, 0.03894456168922911 r,
 0.04175612448730281 r, 0.04466351087439402 r, 0.04766643011428662 r,
 0.05076458191755917 r, 0.0539576564716131 r, 0.05724533447165381 r,
 0.06062728715262111 r, 0.06410317632206519 r, 0.06767265439396564 r,
 0.07133536442348987 r, 0.07509094014268702 r, 0.07893900599711501 r,

```

0.08287917718339499 r, 0.08691105968769186 r, 0.09103425032511492 r,
0.09524833678003664 r, 0.09955289764732322 r, 0.1039475024744748 r,
0.1084317118046711 r, 0.113005077220716 r, 0.1176671413898787 r,
0.1224174381096274 r, 0.1272554923542488 r, 0.1321808203223502 r,
0.1371929294852391 r, 0.1422913186361759 r, 0.1474754779404944 r,
0.152744888986584 r, 0.1580990248377314 r, 0.1635373500848132 r,
0.1690593208998367 r, 0.1746643850903219 r, 0.1803519821545206 r,
0.1861215433374662 r, 0.1919724916878484 r, 0.1979042421157076 r,
0.2039162014509444 r, 0.2100077685026351 r, 0.216178334119151 r,
0.2224272812490723 r, 0.2287539850028937 r, 0.2351578127155118 r,
0.2416381240094921 r, 0.2481942708591053 r, 0.2548255976551299 r,
0.2615314412704124 r, 0.2683111311261794 r, 0.2751639892590951 r,
0.2820893303890569 r, 0.2890864619877229 r, 0.2961546843477643 r,
0.3032932906528349 r, 0.3105015670482534 r, 0.3177787927123868 r,
0.3251242399287333 r, 0.3325371741586922 r, 0.3400168541150183 r,
0.3475625318359485 r, 0.3551734527599992 r, 0.3628488558014202 r,
0.3705879734263036 r, 0.3783900317293359 r, 0.3862542505111889 r,
0.3941798433565377 r, 0.4021660177127022 r, 0.4102119749689023 r,
0.418316910536117 r, 0.4264800139275439 r, 0.4347004688396462 r,
0.4429774532337832 r, 0.451310139418413 r]

```

Berikut kita gambar sikloid untuk $r=1$.

```

>ex &= x-sin(x); ey &= 1-cos(x); aspect(1);
>plot2d(ex,ey,xmin=0,xmax=4pi,square=1); ...
> plot2d("2+cos(x)","1+sin(x)",xmin=0,xmax=2pi,>add,color=blue); ...
> plot2d([2,ex(2)],[1,ey(2)],color=red,>add); ...
> plot2d(ex(2),ey(2),>points,>add,color=red); ...
> plot2d("2pi+cos(x)","1+sin(x)",xmin=0,xmax=2pi,>add,color=blue); ...
> plot2d([2pi,ex(2pi)],[1,ey(2pi)],color=red,>add); ...
> plot2d(ex(2pi),ey(2pi),>points,>add,color=red):

```

Error : [0,1.6666583335744e-7*r-sin(1.6666583335744e-7*r),1.33330666692

```
>ds &= radcan(sqrt(diff(ex,x)^2+diff(ey,x)^2)); $ds=trigsimp(ds) // elemen
```

Maxima said:

```
diff: second argument must be a variable; found errexp1
-- an error. To debug this try: debugmode(true);
```

Error in:

```
ds &= radcan(sqrt(diff(ex,x)^2+diff(ey,x)^2)); $ds=trigsimp(ds ...
^
```

```
>ds &= trigsimp(ds); $ds
```

```
>$showev('integrate(ds,x,0,2*pi)) // hitung panjang sikloid satu putaran pe
```

Maxima said:

```
defint: variable of integration must be a simple or subscripted variable.
defint: found errexp1
#0: showev(f='integrate(ds,[0,1.66665833335744e-7*r,1.33330666692022e-6*r
-- an error. To debug this try: debugmode(true);
```

Error in:

```
$showev('integrate(ds,x,0,2*pi)) // hitung panjang sikloid sat ...
^
```

```
>integrate(mxm("ds"),0,2*pi) // hitung secara numerik
```

Illegal function result in map.

%evalexpression:

```
    if maps then return %mapexpression1(x,f$;args());
```

gauss:

```
    if maps then y=%evalexpression(f$,a+h-(h*xn)',maps;args());
```

adaptivegauss:

```
    t1=gauss(f$,c,c+h;args(),=maps);
```

Try "trace errors" to inspect local variables after errors.

integrate:

```
    return adaptivegauss(f$,a,b,eps*1000;args(),=maps);
```

```
>romberg(mxm("ds"),0,2*pi) // cara lain hitung secara numerik
```

Wrong argument!

Cannot combine a symbolic expression here.
 Did you want to create a symbolic expression?
 Then start with &.

Try "trace errors" to inspect local variables after errors.
 romberg:

```
if cols(y)==1 then return y*(b-a); endif;
Error in:
romberg(mxm("ds"),0,2*pi) // cara lain hitung secara numerik ...
^
```

Perhatikan, seperti terlihat pada gambar, panjang sikloid lebih besar daripada keliling lingkarannya, yakni:

$$2\pi.$$

>

Kurvatur (Kelengkungan) Kurva

image: Osculating.png

Aslinya, kelengkungan kurva diferensiabel (yakni, kurva mulus yang tidak lancip) di titik P didefinisikan melalui lingkaran oskulasi (yaitu, lingkaran yang melalui titik P dan terbaik memperkirakan, paling banyak menyinggung kurva di sekitar P). Pusat dan radius kelengkungan kurva di P adalah pusat dan radius lingkaran oskulasi. Kelengkungan adalah kebalikan dari radius kelengkungan:

$$\kappa = \frac{1}{R}$$

dengan R adalah radius kelengkungan. (Setiap lingkaran memiliki kelengkungan ini pada setiap titiknya, dapat diartikan, setiap lingkaran berputar 2π sejauh $2\pi R$.)

Definisi ini sulit dimanipulasi dan dinyatakan ke dalam rumus untuk kurva umum. Oleh karena itu digunakan definisi lain yang ekivalen.

Definisi Kurvatur dengan Fungsi Parametrik Panjang Kurva

Setiap kurva diferensiabel dapat dinyatakan dengan persamaan parametrik terhadap panjang kurva s:

$$\gamma(s) = (x(s), y(s)),$$

dengan x dan y adalah fungsi riil yang diferensiabel, yang memenuhi:

$$\|\gamma'(s)\| = \sqrt{x'(s)^2 + y'(s)^2} = 1.$$

Ini berarti bahwa vektor singgung

$$\mathbf{T}(s) = (x'(s), y'(s))$$

memiliki norm 1 dan merupakan vektor singgung satuan.

Apabila kurvanya memiliki turunan kedua, artinya turunan kedua x dan y ada, maka $\mathbf{T}'(s)$ ada. Vektor ini merupakan normal kurva yang arahnya menuju pusat kurvatur, norm-nya merupakan nilai kurvatur (kelengkungan):

$$\begin{aligned}\mathbf{T}(s) &= \gamma'(s), \\ \mathbf{T}^2(s) &= 1 \text{ (konstanta)} \Rightarrow \mathbf{T}'(s) \cdot \mathbf{T}(s) = 0 \\ \kappa(s) &= \|\mathbf{T}'(s)\| = \|\gamma''(s)\| = \sqrt{x''(s)^2 + y''(s)^2}.\end{aligned}$$

Nilai

$$R(s) = \frac{1}{\kappa(s)}$$

disebut jari-jari (radius) kelengkungan kurva.

Bilangan riil

$$k(s) = \pm \kappa(s)$$

disebut nilai kelengkungan bertanda.

Contoh:

Akan ditentukan kurvatur lingkaran

$$x = r \cos t, y = r \sin t.$$

```
>fx &= r*cos(t); fy &= r*sin(t);
>&assume(t>0, r>0); s &=integrate(sqrt(diff(fx,t)^2+diff(fy,t)^2),t,0,t); s
```

Maxima said:

```
diff: second argument must be a variable; found errexp1
-- an error. To debug this try: debugmode(true);
```

Error in:

```
... =integrate(sqrt(diff(fx,t)^2+diff(fy,t)^2),t,0,t); s // elemen ...  
^
```

```
>&kill(s); fx &= r*cos(s/r); fy &=r*sin(s/r); // definisi ulang persamaan p
>k &= trigsimp(sqrt(diff(fx,s,2)^2+diff(fy,s,2)^2)); $k // nilai kurvatur l
```

$$\frac{1}{r}$$

Untuk representasi parametrik umum, misalkan

$$x = x(t), y = y(t)$$

merupakan persamaan parametrik untuk kurva bidang yang terdiferensialkan dua kali. Kurvatur untuk kurva tersebut didefinisikan sebagai

$$\begin{aligned} \kappa &= \frac{d\phi}{ds} = \frac{\frac{d\phi}{dt}}{\frac{ds}{dt}} \quad (\phi \text{ adalah sudut kemiringan garis singgung dan } s \text{ adalah panjang kurva}) \\ &= \frac{\frac{d\phi}{dt}}{\sqrt{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2}} = \frac{\frac{d\phi}{dt}}{\sqrt{x'(t)^2 + y'(t)^2}}. \end{aligned}$$

Selanjutnya, pembilang pada persamaan di atas dapat dicari sebagai berikut.

$$\sec^2 \phi \frac{d\phi}{dt} = \frac{d}{dt} (\tan \phi) = \frac{d}{dt} \left(\frac{dy}{dx} \right) = \frac{d}{dt} \left(\frac{dy/dt}{dx/dt} \right) = \frac{d}{dt} \left(\frac{y'(t)}{x'(t)} \right) = \frac{x'(t)y''(t) - x''(t)y'(t)}{x'(t)^2}.$$

$$\begin{aligned} \frac{d\phi}{dt} &= \frac{1}{\sec^2 \phi} \frac{x'(t)y''(t) - x''(t)y'(t)}{x'(t)^2} \\ &= \frac{1}{1 + \tan^2 \phi} \frac{x'(t)y''(t) - x''(t)y'(t)}{x'(t)^2} \\ &= \frac{1}{1 + \left(\frac{y'(t)}{x'(t)}\right)^2} \frac{x'(t)y''(t) - x''(t)y'(t)}{x'(t)^2} \\ &= \frac{x'(t)y''(t) - x''(t)y'(t)}{x'(t)^2 + y'(t)^2}. \end{aligned}$$

Jadi, rumus kurvatur untuk kurva parametrik

$$x = x(t), y = y(t)$$

adalah

$$\kappa(t) = \frac{x'(t)y''(t) - x''(t)y'(t)}{(x'(t)^2 + y'(t)^2)^{3/2}}.$$

Jika kurvanya dinyatakan dengan persamaan parametrik pada koordinat kutub

$$x = r(\theta) \cos \theta, \quad y = r(\theta) \sin \theta,$$

maka rumus kurvturnya adalah

$$\kappa(\theta) = \frac{r(\theta)^2 + 2r'(\theta)^2 - r(\theta)r''(\theta)}{(r'(\theta)^2 + r''(\theta)^2)^{3/2}}.$$

(Silakan Anda turunkan rumus tersebut!)

Contoh:

Lingkaran dengan pusat (0,0) dan jari-jari r dapat dinyatakan dengan persamaan parametrik

$$x = r \cos t, \quad y = r \sin t.$$

Nilai kelengkungan lingkaran tersebut adalah

$$\kappa(t) = \frac{x'(t)y''(t) - x''(t)y'(t)}{(x'(t)^2 + y'(t)^2)^{3/2}} = \frac{r^2}{r^3} = \frac{1}{r}.$$

Hasil cocok dengan definisi kurvatur suatu kelengkungan.

Kurva

$$y = f(x)$$

dapat dinyatakan ke dalam persamaan parametrik

$$x = t, \quad y = f(t), \quad \text{dengan } x'(t) = 1, \quad x''(t) = 0,$$

sehingga kurvturnya adalah

$$\kappa(t) = \frac{y''(t)}{(1 + y'(t)^2)^{3/2}}.$$

Contoh:

Akan ditentukan kurvatur parabola

$$y = ax^2 + bx + c.$$

```
>function f(x) &= a*x^2+b*x+c; $y=f(x)
```

```
>function k(x) &= (diff(f(x),x,2))/(1+diff(f(x),x)^2)^(3/2); $'k(x)=k(x) //
```

Maxima said:

diff: second argument must be a variable; found errexp1
-- an error. To debug this try: debugmode(true);

Error in:

```
... (x) &= (diff(f(x),x,2))/(1+diff(f(x),x)^2)^(3/2); $'k(x)=k(x) ...  
^
```

```
>function f(x) &= x^2+x+1; $y=f(x) // akan kita plot kelengkungan parabola
```

Maxima said:

diff: second argument must be a variable; found errexp1
-- an error. To debug this try: debugmode(true);

Error in:

```
... (x) &= (diff(f(x),x,2))/(1+diff(f(x),x)^2)^(3/2); $'k(x)=k(x) ...  
^
```

Berikut kita gambar parabola tersebut beserta kurva kelengkungan, kurva jari-jari kelengkungan dan salah satu lingkaran oskulasi di titik puncak parabola. Perhatikan, puncak parabola dan jari-jari lingkaran oskulasi di puncak parabola adalah

$$(-1/2, 3/4), \frac{1}{k(2)} = 1/2,$$

sehingga pusat lingkaran oskulasi adalah $(-1/2, 5/4)$.

```
>plot2d(["f(x)", "k(x)"], -2, 1, color=[blue, red]); plot2d("1/k(x)", -1.5, 1, co  
>plot2d("-1/2+1/k(-1/2)*cos(x)", "5/4+1/k(-1/2)*sin(x)", xmin=0, xmax=2pi, >add
```

Error : $f(x)$ does not produce a real or column vector

Error generated by error() command

```
%ploteval:  
    error(f$|" does not produce a real or column vector");  
adaptiveevalone:  
    s=%ploteval(g$,t;args());  
Try "trace errors" to inspect local variables after errors.  
plot2d:  
    dw/n,dw/n^2,dw/n,auto;args());
```

Untuk kurva yang dinyatakan dengan fungsi implisit

$$F(x, y) = 0$$

dengan turunan-turunan parsial

$$F_x = \frac{\partial F}{\partial x}, \quad F_y = \frac{\partial F}{\partial y}, \quad F_{xy} = \frac{\partial}{\partial y} \left(\frac{\partial F}{\partial x} \right), \quad F_{xx} = \frac{\partial}{\partial x} \left(\frac{\partial F}{\partial x} \right), \quad F_{yy} = \frac{\partial}{\partial y} \left(\frac{\partial F}{\partial y} \right),$$

berlaku

$$F_x dx + F_y dy = 0 \text{ atau } \frac{dy}{dx} = -\frac{F_x}{F_y},$$

sehingga kurvturnya adalah

$$\kappa = \frac{F_y^2 F_{xx} - 2F_x F_y F_{xy} + F_x^2 F_{yy}}{(F_x^2 + F_y^2)^{3/2}}.$$

(Silakan Anda turunkan sendiri!)

Contoh 1:

Parabola

$$y = ax^2 + bx + c$$

dapat dinyatakan ke dalam persamaan implisit

$$ax^2 + bx + c - y = 0.$$

```
>function F(x,y) &=a*x^2+b*x+c-y; $F(x,y)
```

```
>Fx &= diff(F(x,y),x), Fxx &=diff(F(x,y),x,2), FY &=diff(F(x,y),y), Fxy &=d
```

Maxima said:

```
diff: second argument must be a variable; found errexp1
-- an error. To debug this try: debugmode(true);
```

Error in:

```
Fx &= diff(F(x,y),x), Fxx &=diff(F(x,y),x,2), Fy &=diff(F(x,y) ...  
^
```

```
>function k(x) &= (Fy^2*Fxx-2*Fx*Fy*Fxy+Fx^2*Fyy)/(Fx^2+Fy^2)^(3/2); $'k(x)
```

$$k([0, 1.66665833335744 \times 10^{-7} r, 1.33330666692022 \times 10^{-6} r, 4.499797504338432 \times 10^{-6} r, 1.066581336583])$$

Hasilnya sama dengan sebelumnya yang menggunakan persamaan parabola biasa. **Lati-han**

- Bukalah buku Kalkulus.
- Cari dan pilih beberapa (paling sedikit 5 fungsi berbeda tipe/bentuk/jenis) fungsi dari buku tersebut, kemudian definisikan di EMT pada baris-baris perintah berikut (jika perlu tambahkan lagi).
- Untuk setiap fungsi, tentukan anti turunannya (jika ada), hitunglah integral tentu dengan batas-batas yang menarik (Anda tentukan sendiri), seperti contoh-contoh tersebut.
- Lakukan hal yang sama untuk fungsi-fungsi yang tidak dapat diintegralkan (cari sedikitnya 3 fungsi).
- Gambar grafik fungsi dan daerah integrasinya pada sumbu koordinat yang sama.
- Gunakan integral tentu untuk mencari luas daerah yang dibatasi oleh dua kurva yang berpotongan di dua titik. (Cari dan gambar kedua kurva dan arsir (warnai) daerah yang dibatasi oleh keduanya.)
- Gunakan integral tentu untuk menghitung volume benda putar kurva $y = f(x)$ yang diputar mengelilingi sumbu x dari $x=a$ sampai $x=b$, yakni

$$V = \int_a^b \pi(f(x)^2 dx.$$

- (Pilih fungsinya dan gambar kurva dan benda putar yang dihasilkan. Anda dapat mencari contoh-contoh bagaimana cara menggambar benda hasil perputaran suatu kurva.)
- Gunakan integral tentu untuk menghitung panjang kurva $y=f(x)$ dari $x=a$ sampai $x=b$ dengan menggunakan rumus:

$$S = \int_a^b \sqrt{1 + (f'(x))^2} dx.$$

(Pilih fungsi dan gambar kurvanya.)

- Apabila fungsi dinyatakan dalam koordinat kutub $x=f(r,t)$, $y=g(r,t)$, $r=h(t)$, $x=a$ bersesuaian dengan $t=t_0$ dan $x=b$ bersesuaian dengan $t=t_1$, maka rumus di atas akan menjadi:

$$S = \int_{t_0}^{t_1} \sqrt{x'(t)^2 + y'(t)^2} dt.$$

- Pilih beberapa kurva menarik (selain lingkaran dan parabola) dari buku kalkulus. Nyatakan setiap kurva tersebut dalam bentuk:

- a. koordinat Kartesius (persamaan $y=f(x)$)
- b. koordinat kutub ($r=r(\theta)$)
- c. persamaan parametrik $x=x(t)$, $y=y(t)$
- d. persamaan implisit $F(x, y)=0$

- Tentukan kurvatur masing-masing kurva dengan menggunakan keempat representasi tersebut (hasilnya harus sama).
- Gambarlah kurva asli, kurva kurvatur, kurva jari-jari lingkaran oskulasi, dan salah satu lingkaran oskulasinya.

>

Barisan dan Deret

(Catatan: bagian ini belum lengkap. Anda dapat membaca contoh-contoh penggunaan EMT dan Maxima untuk menghitung limit barisan, rumus jumlah parsial suatu deret, jumlah tak hingga suatu deret konvergen, dan sebagainya. Anda dapat mengeksplor contoh-contoh di EMT atau perbagai panduan penggunaan Maxima di software Maxima atau dari Internet.)

Barisan dapat didefinisikan dengan beberapa cara di dalam EMT, di antaranya:

- dengan cara yang sama seperti mendefinisikan vektor dengan elemen-elemen beraturan (menggunakan titik dua ":");
- menggunakan perintah "sequence" dan rumus barisan (suku ke $-n$);
- menggunakan perintah "iterate" atau "niterate";
- menggunakan fungsi Maxima "create_list" atau "makelist" untuk menghasilkan barisan simbolik;
- menggunakan fungsi biasa yang inputnya vektor atau barisan;
- menggunakan fungsi rekursif.

EMT menyediakan beberapa perintah (fungsi) terkait barisan, yakni:

- sum: menghitung jumlah semua elemen suatu barisan
- cumsum: jumlah kumulatif suatu barisan
- differences: selisih antar elemen-elemen berturutan

EMT juga dapat digunakan untuk menghitung jumlah deret berhingga maupun deret tak hingga, dengan menggunakan perintah (fungsi) "sum". Perhitungan dapat dilakukan secara numerik maupun simbolik dan eksak.

Berikut adalah beberapa contoh perhitungan barisan dan deret menggunakan EMT.

```
>1:10 // barisan sederhana
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

```
>1:2:30
```

```
[1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29]
```

Iterasi dan Barisan

EMT menyediakan fungsi `iterate("g(x)", x0, n)` untuk melakukan iterasi

$$x_{k+1} = g(x_k), \quad x_0 = x_0, \quad k = 1, 2, 3, \dots, n.$$

Berikut ini disajikan contoh-contoh penggunaan iterasi dan rekursi dengan EMT. Contoh pertama menunjukkan pertumbuhan dari nilai awal 1000 dengan laju pertambahan 5%, selama 10 periode.

```
>q=1.05; iterate("x*q",1000,n=10)'
```

```
1000
1050
1102.5
1157.63
1215.51
1276.28
1340.1
1407.1
1477.46
1551.33
1628.89
```

Contoh berikutnya memperlihatkan bahaya menabung di bank pada masa sekarang! Dengan bunga tabungan sebesar 6% per tahun atau 0.5% per bulan dipotong pajak 20%, dan biaya administrasi 10000 per bulan, tabungan sebesar 1 juta tanpa diambil selama sekitar 10 tahunan akan habis diambil oleh bank!

```
>r=0.005; plot2d(iterate("(1+0.8*r)*x-10000",1000000,n=130)):
```



images/EMT4Kalkulus_Saphira-206.png

Silakan Anda coba-coba, dengan tabungan minimal berapa agar tidak akan habis diambil oleh bank dengan ketentuan bunga dan biaya administrasi seperti di atas.

Berikut adalah perhitungan minimal tabungan agar aman di bank dengan bunga sebesar r dan biaya administrasi a , pajak bunga 20%.

```
>$solve(0.8*r*A-a,A), $% with [r=0.005, a=10]
```

$$\left[A = \frac{5a}{4r} \right]$$

$$[A = 2500.0]$$

Berikut didefinisikan fungsi untuk menghitung saldo tabungan, kemudian dilakukan iterasi.

```
>function saldo(x,r,a) := round((1+0.8*r)*x-a,2);
>iterate({{"saldo",0.005,10}},1000,n=6)
```

```
[1000, 994, 987.98, 981.93, 975.86, 969.76, 963.64]
```

```
>iterate({{"saldo", 0.005, 10}}, 2000, n=6)
```

```
[2000, 1998, 1995.99, 1993.97, 1991.95, 1989.92, 1987.88]
```

```
>iterate({{"saldo", 0.005, 10}}, 2500, n=6)
```

```
[2500, 2500, 2500, 2500, 2500, 2500, 2500]
```

Tabungan senilai 2,5 juta akan aman dan tidak akan berubah nilai (jika tidak ada penarikan), sedangkan jika tabungan awal kurang dari 2,5 juta, lama kelamaan akan berkurang meskipun tidak pernah dilakukan penarikan uang tabungan.

```
>iterate({{"saldo", 0.005, 10}}, 3000, n=6)
```

```
[3000, 3002, 3004.01, 3006.03, 3008.05, 3010.08, 3012.12]
```

Tabungan yang lebih dari 2,5 juta baru akan bertambah jika tidak ada penarikan.

Untuk barisan yang lebih kompleks dapat digunakan fungsi "sequence()". Fungsi ini menghitung nilai-nilai $x[n]$ dari semua nilai sebelumnya, $x[1], \dots, x[n-1]$ yang diketahui.

Berikut adalah contoh barisan Fibonacci.

$$x_n = x_{n-1} + x_{n-2}, \quad x_1 = 1, \quad x_2 = 1$$

```
>sequence("x[n-1]+x[n-2]", [1, 1], 15)
```

```
[1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610]
```

Barisan Fibonacci memiliki banyak sifat menarik, salah satunya adalah akar pangkat ke-n suku ke-n akan konvergen ke pecahan emas:

```
>$(1+sqrt(5))/2=Float((1+sqrt(5))/2)
```

$$\frac{\sqrt{5} + 1}{2} = 1.618033988749895$$

```
>plot2d(sequence("x[n-1]+x[n-2]", [1,1], 250)^(1/(1:250))):
```



images/EMT4Kalkulus_Saphira-211.png

Barisan yang sama juga dapat dihasilkan dengan menggunakan loop.

```
>x=ones(500); for k=3 to 500; x[k]=x[k-1]+x[k-2]; end;
```

Rekursi dapat dilakukan dengan menggunakan rumus yang tergantung pada semua elemen sebelumnya. Pada contoh berikut, elemen ke-n merupakan jumlah (n-1) elemen sebelumnya, dimulai dengan 1 (elemen ke-1). Jelas, nilai elemen ke-n adalah 2^{n-2} , untuk n=2, 4, 5,

```
>sequence("sum(x)", 1, 10)
```

[1, 1, 2, 4, 8, 16, 32, 64, 128, 256]

Selain menggunakan ekspresi dalam x dan n, kita juga dapat menggunakan fungsi. Pada contoh berikut, digunakan iterasi

$$x_n = A \cdot x_{n-1},$$

dengan A suatu matriks 2×2 , dan setiap $x[n]$ merupakan matriks/vektor 2×1 .

```
>A=[1,1;1,2]; function suku(x,n) := A.x[,n-1]
>sequence("suku",[1;1],6)
```

Real 2×6 matrix

1	2	5	13	...
1	3	8	21	...

Hasil yang sama juga dapat diperoleh dengan menggunakan fungsi perpangkatan matriks "matrixpower()". Cara ini lebih cepat, karena hanya menggunakan perkalian matriks sebanyak $\log_2(n)$.

$$x_n = A \cdot x_{n-1} = A^2 \cdot x_{n-2} = A^3 \cdot x_{n-3} = \dots = A^{n-1} \cdot x_1.$$

```
>sequence("matrixpower(A,n).[1;1]",1,6)
```

Real 2×6 matrix

1	5	13	34	...
1	8	21	55	...

Spiral Theodorus

image: Spiral_of_Theodorus.png

Spiral Theodorus (spiral segitiga siku-siku) dapat digambar secara rekursif. Rumus rekursifnya adalah:

$$x_n = \left(1 + \frac{i}{\sqrt{n-1}}\right) x_{n-1}, \quad x_1 = 1,$$

yang menghasilkan barisan bilangan kompleks.

```
>function g(n) := 1+I/sqrt(n)
```

Rekursinya dapat dijalankan sebanyak 17 untuk menghasilkan barisan 17 bilangan kompleks, kemudian digambar bilangan-bilangan kompleksnya.

```
>x=sequence("g(n-1)*x[n-1]",1,17); plot2d(x,r=3.5); textbox(latex("Spiral\
```



images/EMT4Kalkulus_Saphira-215.png

A large rectangular frame contains a spiral pattern of points plotted in the complex plane. The spiral starts at the origin (0,0) and moves outwards, with each subsequent point being the product of the previous point and a constant factor. The points form a logarithmic spiral that fills a circular region of radius approximately 3.5 units.

Selanjutnya dihubungkan titik 0 dengan titik-titik kompleks tersebut menggunakan loop.

```
>for i=1:cols(x); plot2d([0,x[i]],>add); end:
```



images/EMT4Kalkulus_Saphira-216.png

>

Spiral tersebut juga dapat didefinisikan menggunakan fungsi rekursif, yang tidak memerlukan indeks dan bilangan kompleks. Dalam hal ini diigunakan vektor kolom pada bidang.

```
>function gstep (v) ...
```

```
w=[-v[2];v[1]];
return v+w/norm(w);
endfunction
```

Jika dilakukan iterasi 16 kali dimulai dari [1;0] akan didapatkan matriks yang memuat vektor-vektor dari setiap iterasi.

```
>x=iterate("gstep", [1;0],16); plot2d(x[1],x[2],r=3.5,>points):
```



images/EMT4Kalkulus_Saphira-217.png

Kekonvergenan

Terkadang kita ingin melakukan iterasi sampai konvergen. Apabila iterasinya tidak konvergen setelah ditunggu lama, Anda dapat menghentikannya dengan menekan tombol [ESC].

```
>iterate("cos(x)",1) // iterasi x(n+1)=cos(x(n)), dengan x(0)=1.
```

0.739085133216

Iterasi tersebut konvergen ke penyelesaian persamaan

$$x = \cos(x).$$

Iterasi ini juga dapat dilakukan pada interval, hasilnya adalah barisan interval yang memuat akar tersebut.

```
>hasil := iterate("cos(x)",~1,2~) //iterasi x(n+1)=cos(x(n)), dengan interval
```

~0.739085133211, 0.7390851332133~

Jika interval hasil tersebut sedikit diperlebar, akan terlihat bahwa interval tersebut memuat akar persamaan $x=\cos(x)$.

```
>h=expand(hasil,100), cos(h) << h
```

```
~0.73908513309, 0.73908513333~  
1
```

Iterasi juga dapat digunakan pada fungsi yang didefinisikan.

```
>function f(x) := (x+2/x)/2
```

Iterasi $x(n+1)=f(x(n))$ akan konvergen ke akar kuadrat 2.

```
>iterate("f",2), sqrt(2)
```

```
1.41421356237  
1.41421356237
```

Jika pada perintah iterate diberikan tambahan parameter n, maka hasil iterasinya akan ditampilkan mulai dari iterasi pertama sampai ke-n.

```
>iterate("f",2,5)
```

```
[2, 1.5, 1.41667, 1.41422, 1.41421, 1.41421]
```

Untuk iterasi ini tidak dapat dilakukan terhadap interval.

```
>niterate("f",~1,2~,5)
```

```
[ ~1,2~, ~1,2~, ~1,2~, ~1,2~, ~1,2~, ~1,2~ ]
```

Perhatikan, hasil iterasinya sama dengan interval awal. Alasannya adalah perhitungan dengan interval bersifat terlalu longgar. Untuk meningkatkan perhitungan pada ekspresi dapat digunakan pembagian intervalnya, menggunakan fungsi ieval().

```
>function s(x) := ieval("(x+2/x)/2",x,10)
```

Selanjutnya dapat dilakukan iterasi hingga diperoleh hasil optimal, dan intervalnya tidak semakin mengecil. Hasilnya berupa interval yang memuat akar persamaan:

$$x = \frac{1}{2} \left(x + \frac{2}{x} \right).$$

Satu-satunya solusi adalah

$$x = \sqrt{2}.$$

```
>iterate("s", ~1, 2~)
```

```
~1.41421356236, 1.41421356239~
```

Fungsi "iterate()" juga dapat bekerja pada vektor. Berikut adalah contoh fungsi vektor, yang menghasilkan rata-rata aritmetika dan rata-rata geometri.

$$(a_{n+1}, b_{n+1}) = \left(\frac{a_n + b_n}{2}, \sqrt{a_n b_n} \right)$$

Iterasi ke-n disimpan pada vektor kolom x[n].

```
>function g(x) := [(x[1]+x[2])/2; sqrt(x[1]*x[2])]
```

Iterasi dengan menggunakan fungsi tersebut akan konvergen ke rata-rata aritmetika dan geometri dari nilai-nilai awal.

```
>iterate("g", [1; 5])
```

```
2.60401
```

```
2.60401
```

Hasil tersebut konvergen agak cepat, seperti kita cek sebagai berikut.

```
>iterate("g", [1; 5], 4)
```

1	3	2.61803	2.60403	2.60401
5	2.23607	2.59002	2.60399	2.60401

Iterasi pada interval dapat dilakukan dan stabil, namun tidak menunjukkan bahwa limitnya pada batas-batas yang dihitung.

```
>iterate("g", [~1~; ~5~], 4)
```

Interval 2 x 5 matrix

~0.99999999999999778, 1.0000000000000022~ ...
 ~4.999999999999911, 5.0000000000000089~ ...

Iterasi berikut konvergen sangat lambat.

$$x_{n+1} = \sqrt{x_n}.$$

```
>iterate("sqrt(x)", 2, 10)
```

[2, 1.41421, 1.18921, 1.09051, 1.04427, 1.0219, 1.01089,
 1.00543, 1.00271, 1.00135, 1.00068]

Kekonvergenan iterasi tersebut dapat dipercepat dengan percepatan Steffenson:

```
>steffenson("sqrt(x)", 2, 10)
```

[1.04888, 1.00028, 1, 1]

Iterasi menggunakan Loop yang ditulis Langsung

Berikut adalah beberapa contoh penggunaan loop untuk melakukan iterasi yang ditulis langsung pada baris perintah.

```
>x=2; repeat x=(x+2/x)/2; until x^2~=2; end; x,
```

1.41421356237

Penggabungan matriks menggunakan tanda "|" dapat digunakan untuk menyimpan semua hasil iterasi.

```
>v=[1]; for i=2 to 8; v=v| (v[i-1]*i); end; v,
```

[1, 2, 6, 24, 120, 720, 5040, 40320]

hasil iterasi juga dapat disimpan pada vektor yang sudah ada.

```
>v=ones(1,100); for i=2 to cols(v); v[i]=v[i-1]*i; end; ...
>plot2d(v,logplot=1); textbox(latex(&log(n)),x=0.5):
```

images/EMT4Kalkulus_Saphira-223.png

```
>A =[0.5,0.2;0.7,0.1]; b=[2;2]; ...
>x=[1;1]; repeat xnew=A.x-b; until all(xnew~≈x); x=xnew; end; ...
>x,
```

-7.09677

-7.74194

Iterasi di dalam Fungsi

Fungsi atau program juga dapat menggunakan iterasi dan dapat digunakan untuk melakukan iterasi. Berikut adalah beberapa contoh iterasi di dalam fungsi.

Contoh berikut adalah suatu fungsi untuk menghitung berapa lama suatu iterasi konvergen. Nilai fungsi tersebut adalah hasil akhir iterasi dan banyak iterasi sampai konvergen.

```
>function map hiter(f$,x0) ...
```

```
x=x0;
maxiter=0;
repeat
    xnew=f$(x);
    maxiter=maxiter+1;
    until xnew~=x;
    x=xnew;
end;
return maxiter;
endfunction
```

Misalnya, berikut adalah iterasi untuk mendapatkan hampiran akar kuadrat 2, cukup cepat, konvergen pada iterasi ke-5, jika dimulai dari hampiran awal 2.

```
>hiter("(x+2/x)/2", 2)
```

5

Karena fungsinya didefinisikan menggunakan "map". maka nilai awalnya dapat berupa vektor.

```
>x=1.5:0.1:10; hasil=hiter("(x+2/x)/2", x); ...
> plot2d(x,hasil);
```

images/EMT4Kalkulus_Saphira-224.png

Dari gambar di atas terlihat bahwa kekonvergenan iterasinya semakin lambat, untuk nilai awal semakin besar, namun penambahannya tidak kontinu. Kita dapat menemukan kapan maksimum iterasinya bertambah.

```
>hasil[1:10]
```

```
[4, 5, 5, 5, 5, 5, 6, 6, 6]
```

```
>x[nonzeros(differences(hasil))]
```

```
[1.5, 2, 3.4, 6.6]
```

maksimum iterasi sampai konvergen meningkat pada saat nilai awalnya 1.5, 2, 3.4, dan 6.6. Contoh berikutnya adalah metode Newton pada polinomial kompleks berderajat 3.

```
>p &= x^3-1; newton &= x-p/diff(p,x); $newton
```

Maxima said:

```
diff: second argument must be a variable; found errexp1
-- an error. To debug this try: debugmode(true);
```

Error in:

```
p &= x^3-1; newton &= x-p/diff(p,x); $newton ...  
^
```

Selanjutnya didefinisikan fungsi untuk melakukan iterasi (aslanya 10 kali).

```
>function iterasi(f$,x,n=10) ...
```

```
loop 1 to n; x=f$(x); end;
return x;
endfunction
```

Kita mulai dengan menentukan titik-titik grid pada bidang kompleksnya.

```
>r=1.5; x=linspace(-r,r,501); Z=x+I*x'; W=iterasi(newton,Z);
```

Function newton needs at least 3 arguments!

Use: newton (f\$: call, df\$: call, x: scalar complex {, y: number, eps: no

Error in:

```
... x=linspace(-r,r,501); Z=x+I*x'; W=iterasi(newton,Z); ...  
^
```

Berikut adalah akar-akar polinomial di atas.

```
>z=&solve(p) ()
```

```
Maxima said:  
solve: more equations than unknowns.  
Unknowns given :  
[r]  
Equations given:  
errexpl  
-- an error. To debug this try: debugmode(true);  
  
Error in:  
z=&solve(p) () ...  
^
```

Untuk menggambar hasil iterasinya, dihitung jarak dari hasil iterasi ke-10 ke masing-masing akar, kemudian digunakan untuk menghitung warna yang akan digambar, yang menunjukkan limit untuk masing-masing nilai awal.

Fungsi plotrgb() menggunakan jendela gambar terkini untuk menggambar warna RGB sebagai matriks.

```
>C=rgb(max(abs(W-z[1]),1),max(abs(W-z[2]),1),max(abs(W-z[3]),1)); ...  
> plot2d(None,-r,r,-r,r); plotrgb(C):
```

```
Variable W not found!  
Error in:  
C=rgb(max(abs(W-z[1]),1),max(abs(W-z[2]),1),max(abs(W-z[3]),1)) ...  
^
```

Iterasi Simbolik

Seperti sudah dibahas sebelumnya, untuk menghasilkan barisan ekspresi simbolik dengan Maxima dapat digunakan fungsi makelist().

```
>&powerdisp:true // untuk menampilkan deret pangkat mulai dari suku berpang
```

true

```
>deret &= makelist(taylor(exp(x),x,0,k),k,1,3); $deret // barisan deret Tay
```

Maxima said:

```
taylor: 0.1539740213994798*r cannot be a variable.  
-- an error. To debug this try: debugmode(true);
```

Error in:

```
deret &= makelist(taylor(exp(x),x,0,k),k,1,3); $deret // baris ...  
^
```

Untuk mengubah barisan deret tersebut menjadi vektor string di EMT digunakan fungsi mxm2str(). Selanjutnya, vektor string/ekspresi hasilnya dapat digambar seperti menggambar vektor ekspresi pada EMT.

```
>plot2d("exp(x)",0,3); // plot fungsi aslinya, e^x
```

```
>plot2d(mxm2str("deret"),>add,color=4:6): // plot ketiga deret taylor hampi
```

Maxima said:

```
length: argument cannot be a symbol; found deret  
-- an error. To debug this try: debugmode(true);
```

mxmeval:

```
    return evaluate(mxm(s));
```

Try "trace errors" to inspect local variables after errors.

mxm2str:

```
n=mxmeval("length(VVV)");
```

Selain cara di atas dapat juga dengan cara menggunakan indeks pada vektor/list yang dihasilkan.

```
>$deret[3]
```

*deret*₃

```
>plot2d(["exp(x)",&deret[1],&deret[2],&deret[3]],0,3,color=1:4):
```

deret is not a variable!

Error in expression: deret[1]

%ploteval:

```
y0=f$(x[1],args());
```

Try "trace errors" to inspect local variables after errors.

plot2d:

```
u=u_(%ploteval(xx[],t;args()));
```

```
>$sum(sin(k*x)/k,k,1,5)
```



Berikut adalah cara menggambar kurva

$$y = \sin(x) + \frac{\sin 3x}{3} + \frac{\sin 5x}{5} + \dots$$

```
>plot2d(&sum(sin((2*k+1)*x)/(2*k+1),k,0,20),0,2pi):
```

Maxima output too long!

Error in:

```
plot2d(&sum(sin((2*k+1)*x)/(2*k+1),k,0,20),0,2pi): ...  
^
```

Hal serupa juga dapat dilakukan dengan menggunakan matriks, misalkan kita akan menggambar kurva

$$y = \sum_{k=1}^{100} \frac{\sin(kx)}{k}, \quad 0 \leq x \leq 2\pi.$$

```
>x=linspace(0,2pi,1000); k=1:100; y=sum(sin(k*x')/k)'; plot2d(x,y):
```



images/EMT4Kalkulus_Saphira-229.png

Tabel Fungsi

Terdapat cara menarik untuk menghasilkan barisan dengan ekspresi Maxima. Perintah mxmttable() berguna untuk menampilkan dan menggambar barisan dan menghasilkan barisan sebagai vektor kolom.

Sebagai contoh berikut adalah barisan turunan ke-n x^x di $x=1$.

```
>mxmttable("diffat(x^x,x=1,n)","n",1,8,frac=1);
```

Maxima said:

```
diff: second argument must be a variable; found errexp1
#0: diffat(expr=[0,1.66665833335744e-7*r,1.33330666692022e-6*r,4.49979750
-- an error. To debug this try: debugmode(true);
```

%mxmevtable:

```
    return mxm("@expr,@var=@value")();
```

Try "trace errors" to inspect local variables after errors.

mxmttable:

```
y[#,1]=%mxmevtable(expr,var,x[#]);
```

```
>$' sum(k, k, 1, n) = factor(ev(sum(k, k, 1, n),simpsum=true)) // simpsum:me
```

$$\sum_{k=1}^n k = \frac{n(1+n)}{2}$$

```
> $' sum(1/(3^k+k), k, 0, inf) = factor(ev(sum(1/(3^k+k), k, 0, inf), simpsum=
```

$$\sum_{k=0}^{\infty} \frac{1}{k+3^k} = \sum_{k=0}^{\infty} \frac{1}{k+3^k}$$

Di sini masih gagal, hasilnya tidak dihitung.

```
> $' sum(1/x^2, x, 1, inf) = ev(sum(1/x^2, x, 1, inf), simpsum=true) // ev: men
```

$$\sum_{x=1}^{\infty} \frac{1}{x^2} = \frac{\pi^2}{6}$$

```
> $' sum((-1)^(k-1)/k, k, 1, inf) = factor(ev(sum((-1)^(x-1)/x, x, 1, inf), si
```

$$\sum_{k=1}^{\infty} \frac{(-1)^{-1+k}}{k} = - \sum_{x=1}^{\infty} \frac{(-1)^x}{x}$$

Di sini masih gagal, hasilnya tidak dihitung.

```
> $' sum((-1)^k/(2*k-1), k, 1, inf) = factor(ev(sum((-1)^k/(2*k-1), k, 1, inf)
```

$$\sum_{k=1}^{\infty} \frac{(-1)^k}{-1+2k} = \sum_{k=1}^{\infty} \frac{(-1)^k}{-1+2k}$$

```
> $ev(sum(1/n!, n, 0, inf), simpsum=true)
```

$$\sum_{n=0}^{\infty} \frac{1}{n!}$$

Di sini masih gagal, hasilnya tidak dihitung, harusnya hasilnya e.

```
>&assume(abs(x)<1); $' sum(a*x^k, k, 0, inf)=ev(sum(a*x^k, k, 0, inf)), simpsum=true;
```

Answering "Is $-94914474571+15819\pi r$ positive, negative or zero?" with "positive". Maxima said:

sum: sum is divergent.
-- an error. To debug this try: debugmode(true);

Error in:
... k, 0, inf)=ev(sum(a*x^k, k, 0, inf), simpsum=true), &forget(abs ...
^

Deret geometri tak hingga, dengan asumsi rasional antara -1 dan 1.

```
>$' sum(x^k/k!, k, 0, inf)=ev(sum(x^k/k!, k, 0, inf), simpsum=true)
```

$$\left[0, \sum_{k=0}^{\infty} \frac{(1.66665833335744 \times 10^{-7})^k r^k}{k!}, \sum_{k=0}^{\infty} \frac{(1.33330666692022 \times 10^{-6})^k r^k}{k!}, \sum_{k=0}^{\infty} \frac{(4.499797504338432 \times 10^{-1})^k r^k}{k!} \right]$$

```
>$limit(sum(x^k/k!, k, 0, n), n, inf)
```

$$\left[0, \lim_{n \rightarrow \infty} \sum_{k=0}^n \frac{(1.66665833335744 \times 10^{-7})^k r^k}{k!}, \lim_{n \rightarrow \infty} \sum_{k=0}^n \frac{(1.33330666692022 \times 10^{-6})^k r^k}{k!}, \lim_{n \rightarrow \infty} \sum_{k=0}^n \frac{(4.499797504338432 \times 10^{-1})^k r^k}{k!} \right]$$

```
>function d(n) &= sum(1/(k^2-k), k, 2, n); $' d(n)=d(n)
```

$$d(n) = \sum_{k=2}^n \frac{1}{-k + k^2}$$

```
>$d(10)=ev(d(10), simpsum=true)
```

$$\sum_{k=2}^{10} \frac{1}{-k + k^2} = \frac{9}{10}$$

```
>$d(100)=ev(d(100), simpsum=true)
```

$$\sum_{k=2}^{100} \frac{1}{-k + k^2} = \frac{99}{100}$$

>

Deret Taylor

Deret Taylor suatu fungsi f yang diferensiabel sampai tak hingga di sekitar $x=a$ adalah:

$$f(x) = \sum_{k=0}^{\infty} \frac{(x-a)^k f^{(k)}(a)}{k!}.$$

```
> $' e^x =taylor(exp(x),x,0,10) // deret Taylor e^x di sekitar x=0, sampai su
```

Maxima said:

```
taylor: 0.1539740213994798*r cannot be a variable.  
-- an error. To debug this try: debugmode(true);
```

Error in:

```
$' e^x =taylor(exp(x),x,0,10) // deret Taylor e^x di sekitar x= ...  
^
```

```
> $' log(x)=taylor(log(x),x,1,10) // deret log(x) di sekitar x=1
```

Maxima said:

```
log: encountered log(0).  
-- an error. To debug this try: debugmode(true);
```

Error in:

```
$' log(x)=taylor(log(x),x,1,10) // deret log(x) di sekitar x=1 ...  
^
```


BAB

Visualisasi dan Perhitungan Geometri dengan EMT

Euler menyediakan beberapa fungsi untuk melakukan visualisasi dan perhitungan geometri, baik secara numerik maupun analitik (seperti biasanya tentunya, menggunakan Maxima). Fungsi-fungsi untuk visualisasi dan perhitungan geometri tersebut disimpan di dalam file program "geometry.e", sehingga file tersebut harus dipanggil sebelum menggunakan fungsi-fungsi atau perintah-perintah untuk geometri.

```
>load geometry
```

Numerical and symbolic geometry.

Fungsi-fungsi Geometri

Fungsi-fungsi untuk Menggambar Objek Geometri:

```
defaultd:=textheight()*1.5: nilai asli untuk parameter d
setPlotrange(x1,x2,y1,y2): menentukan rentang x dan y pada bidang koordinat
setPlotRange(r): pusat bidang koordinat (0,0) dan batas-batas sumbu-x dan y
plotPoint (P, "P"): menggambar titik P dan diberi label "P"
plotSegment (A,B, "AB", d): menggambar ruas garis AB, diberi label "AB" sejauh d
plotLine (g, "g", d): menggambar garis g diberi label "g" sejauh d
plotCircle (c,"c",v,d): Menggambar lingkaran c dan diberi label "c"
plotLabel (label, P, V, d): menuliskan label pada posisi P
```

Fungsi-fungsi Geometri Analitik (numerik maupun simbolik):

310 BAB 6. VISUALISASI DAN PERHITUNGAN GEOMETRI DENGAN EMT

```
turn(v, phi): memutar vektor v sejauh phi
turnLeft(v): memutar vektor v ke kiri
turnRight(v): memutar vektor v ke kanan
normalize(v): normal vektor v
crossProduct(v, w): hasil kali silang vektorv dan w.
lineThrough(A, B): garis melalui A dan B, hasilnya [a,b,c] sdh. ax+by=c.
lineWithDirection(A,v): garis melalui A searah vektor v
getLineDirection(g): vektor arah (gradien) garis g
getNormal(g): vektor normal (tegak lurus) garis g
getPointOnLine(g): titik pada garis g
perpendicular(A, g): garis melalui A tegak lurus garis g
parallel (A, g): garis melalui A sejajar garis g
lineIntersection(g, h): titik potong garis g dan h
projectToLine(A, g): proyeksi titik A pada garis g
distance(A, B): jarak titik A dan B
distanceSquared(A, B): kuadrat jarak A dan B
quadrance(A, B): kuadrat jarak A dan B
areaTriangle(A, B, C): luas segitiga ABC
computeAngle(A, B, C): besar sudut <ABC
angleBisector(A, B, C): garis bagi sudut <ABC
circleWithCenter (A, r): lingkaran dengan pusat A dan jari-jari r
getCircleCenter(c): pusat lingkaran c
getCircleRadius(c): jari-jari lingkaran c
circleThrough(A,B,C): lingkaran melalui A, B, C
middlePerpendicular(A, B): titik tengah AB
lineCircleIntersections(g, c): titik potong garis g dan lingkaran c
circleCircleIntersections (c1, c2): titik potong lingkaran c1 dan c2
planeThrough(A, B, C): bidang melalui titik A, B, C
```

Fungsi-fungsi Khusus Untuk Geometri Simbolik:

```
getLineEquation (g,x,y): persamaan garis g dinyatakan dalam x dan y
getHesseForm (g,x,y,A): bentuk Hesse garis g dinyatakan dalam x dan y dengan sisi positif (kanan/atasi) garis
quad(A,B): kuadrat jarak AB
spread(a,b,c): Spread segitiga dengan panjang sisi-sisi a,b,c, yakni sin(alpha)/sisi yang menghadap sisi a.
crosslaw(a,b,c,sa): persamaan 3 quads dan 1 spread pada segitiga dengan p
triplespread(sa,sb,sc): persamaan 3 spread sa,sb,sc yang membentuk suatu
doublespread(sa): Spread sudut rangkap Spread 2*phi, dengan sa=sin(phi)^2
```

Contoh 1: Luas, Lingkaran Luar, Lingkaran Dalam Segitiga

Untuk menggambar objek-objek geometri, langkah pertama adalah menentukan rentang sumbu-sumbu koordinat. Semua objek geometri akan digambar pada satu bidang koordinat, sampai didefinisikan bidang koordinat yang baru.

```
>setPlotRange (-0.5,2.5,-0.5,2.5); // mendefinisikan bidang koordinat baru
```

Sekarang tetapkan tiga titik dan gambarkan.

```
>A=[1,0]; plotPoint(A,"A"); // definisi dan gambar tiga titik
>B=[0,1]; plotPoint(B,"B");
>C=[2,2]; plotPoint(C,"C");
```

Lalu 3 bagian.

```
>plotSegment(A,B,"c"); // c=AB
>plotSegment(B,C,"a"); // a=BC
>plotSegment(A,C,"b"); // b=AC
```

Fungsi geometri meliputi fungsi untuk membuat garis dan lingkaran. Format barisnya adalah [a,b,c], yang mewakili baris dengan persamaan $ax+by=c$.

```
>lineThrough(B,C) // garis yang melalui B dan C
```

$[-1, 2, 2]$

Hitung garis tegak lurus yang melalui A di BC.

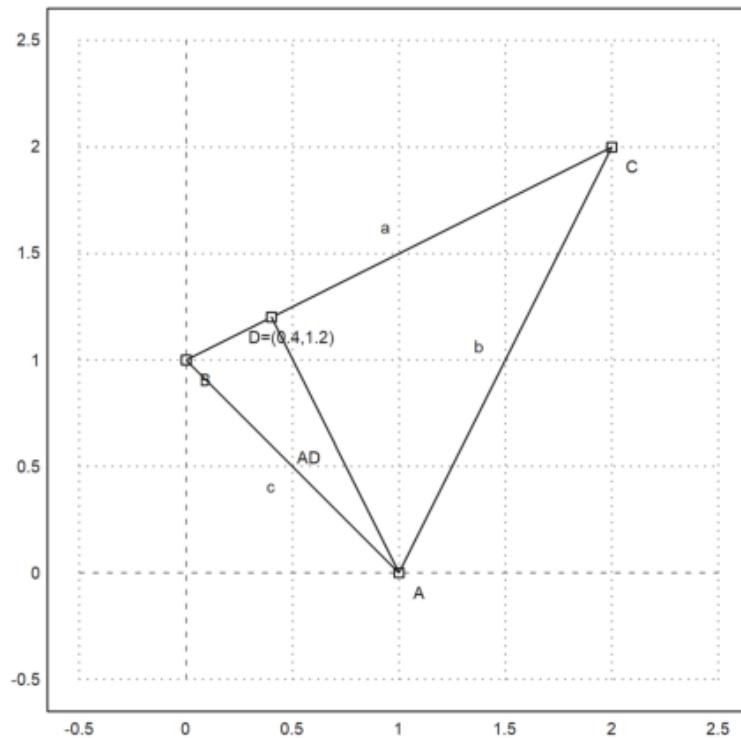
```
>h=perpendicular(A,lineThrough(B,C)); // garis h tegak lurus BC melalui A
```

Dan simpangannya dengan BC.

```
>D=lineIntersection(h,lineThrough(B,C)); // D adalah titik potong h dan BC
```

Buatlah plotnya.

```
>plotPoint(D,value=1); // koordinat D ditampilkan
>aspect(1); plotSegment(A,D); // tampilkan semua gambar hasil plot...()
```



Hitung luas ABC:

$$L_{\triangle ABC} = \frac{1}{2} AD \cdot BC.$$

```
>norm(A-D)*norm(B-C)/2 // AD=norm(A-D), BC=norm(B-C)
```

1.5

Bandingkan dengan rumus determinan.

```
>areaTriangle(A,B,C) // hitung luas segitiga langsung dengan fungsi
```

1.5

Cara lain menghitung luas segitiga ABC:

```
>distance(A,D)*distance(B,C)/2
```

1.5

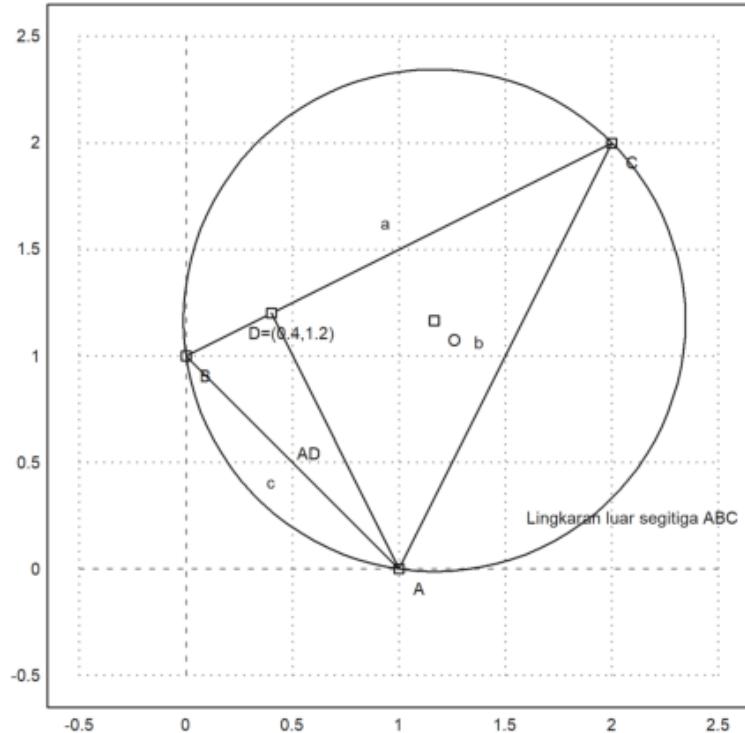
Sudutnya di C.

```
>degsprint(computeAngle(B,C,A))
```

$36^\circ 52' 11.63''$

Sekarang lingkaran luar segitiga.

```
>c=circleThrough(A,B,C); // lingkaran luar segitiga ABC
>R=getCircleRadius(c); // jari2 lingkaran luar
>O=getCircleCenter(c); // titik pusat lingkaran c
>plotPoint(O,"O"); // gambar titik "O"
>plotCircle(c,"Lingkaran luar segitiga ABC"):
```



Tampilkan koordinat titik pusat dan jari-jari lingkaran luar.

```
>O, R
```

[1.16667, 1.16667]

1.17851130198

Sekarang akan digambar lingkaran dalam segitiga ABC. Titik pusat lingkaran dalam adalah titik potong garis-garis bagi sudut.

```
>l=angleBisector(A,C,B); // garis bagi <ACB
>g=angleBisector(C,A,B); // garis bagi <CAB
>P=lineIntersection(l,g) // titik potong kedua garis bagi sudut
```

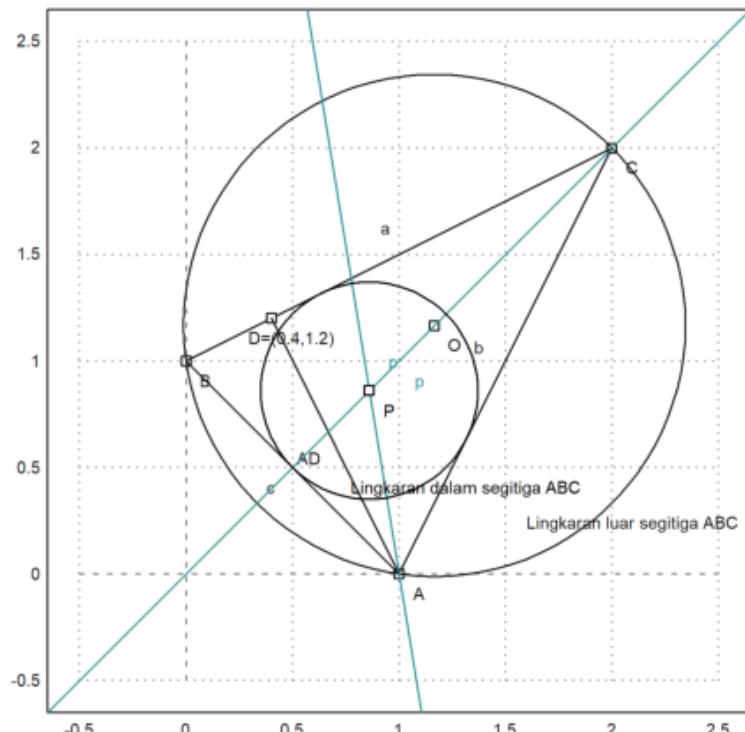
[0.86038, 0.86038]

Tambahkan apapun ke dalam plot

```
>color(5); plotLine(l); plotLine(g); color(1); // gambar kedua garis bagi s
>plotPoint(P, "P"); // gambar titik potongnya
>r=norm(P-projectToLine(P, lineThrough(A,B))) // jari-jari lingkaran dalam
```

0.509653732104

```
>plotCircle(circleWithCenter(P,r), "Lingkaran dalam segitiga ABC"); // gamba
```



1. Tentukan ketiga titik singgung lingkaran dalam dengan sisi-sisi segitiga ABC.
2. Gambar segitiga dengan titik-titik sudut ketiga titik singgung tersebut. Merupakan segitiga apakah itu?
3. Hitung luas segitiga tersebut.
4. Tunjukkan bahwa garis bagi sudut yang ke tiga juga melalui titik pusat lingkaran dalam.
5. Gambar jari-jari lingkaran dalam.
6. Hitung luas lingkaran luar dan luas lingkaran dalam segitiga ABC. Adakah hubungan antara luas kedua lingkaran tersebut dengan luas segitiga ABC?

Contoh 2: Geometri Simbolik

Kita dapat menghitung geometri eksak dan simbolik menggunakan Maxima.

File geometry.e menyediakan fungsi yang sama (dan lebih banyak lagi) di Maxima. Namun, sekarang kita dapat menggunakan perhitungan simbolik.

```
>A &= [1,0]; B &= [0,1]; C &= [2,2]; // menentukan tiga titik A, B, C
```

Fungsi garis dan lingkaran berfungsi sama seperti fungsi Euler, namun menyediakan komputasi simbolis.

```
>c &= lineThrough(B,C) // c=BC
```

$[-1, 2, 2]$

Kita bisa mendapatkan persamaan garis dengan mudah.

```
>$getLineEquation(c,x,y), $solve(%,y) | expand // persamaan garis c
```

$$2y - x = 2$$

$$\left[y = \frac{x}{2} + 1 \right]$$

```
>$getLineEquation(lineThrough([x1,y1], [x2,y2]),x,y), $solve(%,y) // persamaan garis
```

BAB 6. VISUALISASI DAN PERHITUNGAN GEOMETRI DENGAN EMT

$$x(y_1 - y_2) + (x_2 - x_1) y = x_1(y_1 - y_2) + (x_2 - x_1) y_1$$

$$\left[y = \frac{-(x_1 - x) y_2 - (x - x_2) y_1}{x_2 - x_1} \right]$$

```
>$getLineEquation(lineThrough(A,[x1,y1]),x,y) // persamaan garis melalui A
```

$$(x_1 - 1) y - x y_1 = -y_1$$

```
>h &= perpendicular(A,lineThrough(B,C)) // h melalui A tegak lurus BC
```

$$[2, 1, 2]$$

```
>Q &= lineIntersection(c,h) // Q titik potong garis c=BC dan h
```

$$\begin{matrix} 2 & 6 \\ [-, -] \\ 5 & 5 \end{matrix}$$

```
>$projectToLine(A,lineThrough(B,C)) // proyeksi A pada BC
```

$$\left[\frac{2}{5}, \frac{6}{5} \right]$$

```
>$distance(A,Q) // jarak AQ
```

$$\frac{3}{\sqrt{5}}$$

```
>cc &= circleThrough(A,B,C); $cc // (titik pusat dan jari-jari) lingkaran m
```

$$\left[\frac{7}{6}, \frac{7}{6}, \frac{5}{3\sqrt{2}} \right]$$

```
>r=>getCircleRadius(cc); $r , $float(r) // tampilan nilai jari-jari
```

$$\frac{5}{3\sqrt{2}}$$

1.178511301977579

```
>$computeAngle(A,C,B) // nilai <ACB
```

$$\arccos\left(\frac{4}{5}\right)$$

```
>$solve(getLineEquation(angleBisector(A,C,B),x,y),y)[1] // persamaan garis
```

$$y = x$$

```
>P &= lineIntersection(angleBisector(A,C,B),angleBisector(C,B,A)); $P // ti
```

$$\left[\frac{\sqrt{2}\sqrt{5}+2}{6}, \frac{\sqrt{2}\sqrt{5}+2}{6} \right]$$

```
>P() // hasilnya sama dengan perhitungan sebelumnya
```

[0.86038, 0.86038]

Perpotongan Garis dan Lingkaran

Tentu saja, kita juga bisa memotong garis dengan lingkaran, dan lingkaran dengan lingkaran.

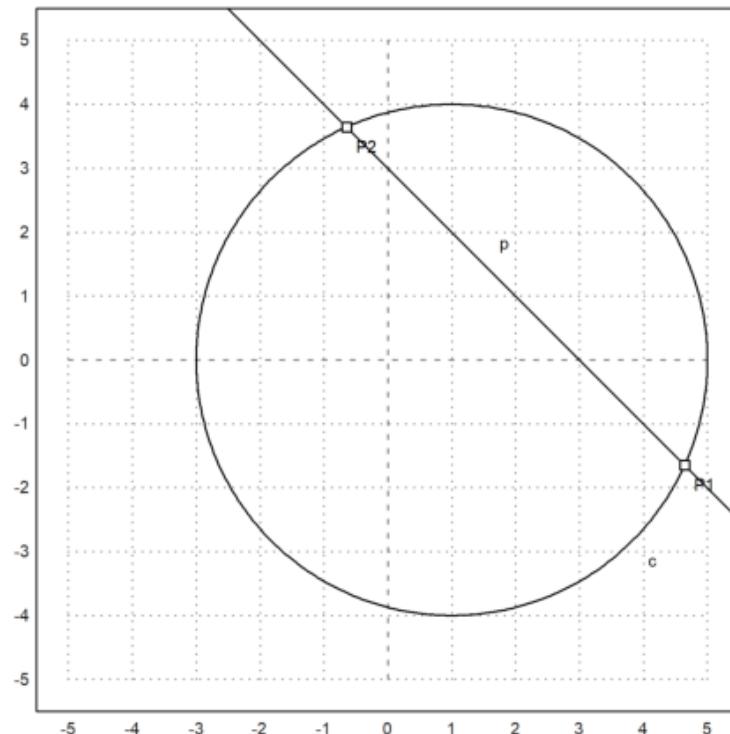
```
>A &:= [1, 0]; c=circleWithCenter(A, 4);
>B &:= [1, 2]; C &:= [2, 1]; l=lineThrough(B,C);
>setPlotRange(5); plotCircle(c); plotLine(l);
```

Perpotongan garis dengan lingkaran menghasilkan dua titik dan jumlah titik perpotongan.

```
>{P1,P2,f}=lineCircleIntersections(l,c);
>P1, P2, f
```

```
[4.64575, -1.64575]
[-0.645751, 3.64575]
2
```

```
>plotPoint(P1); plotPoint(P2);
```



Hal yang sama di Maxima.

```
>c &= circleWithCenter(A,4) // lingkaran dengan pusat A jari-jari 4
```

[1, 0, 4]

```
>l &= lineThrough(B,C) // garis l melalui B dan C
```

[1, 1, 3]

```
>$lineCircleIntersections(l,c) | radcan, // titik potong lingkaran c dan ga
```

$$\left[\left[\sqrt{7} + 2, 1 - \sqrt{7} \right], \left[2 - \sqrt{7}, \sqrt{7} + 1 \right] \right]$$

Akan ditunjukkan bahwa sudut-sudut yang menghadap bsuusr yang sama adalah sama besar.

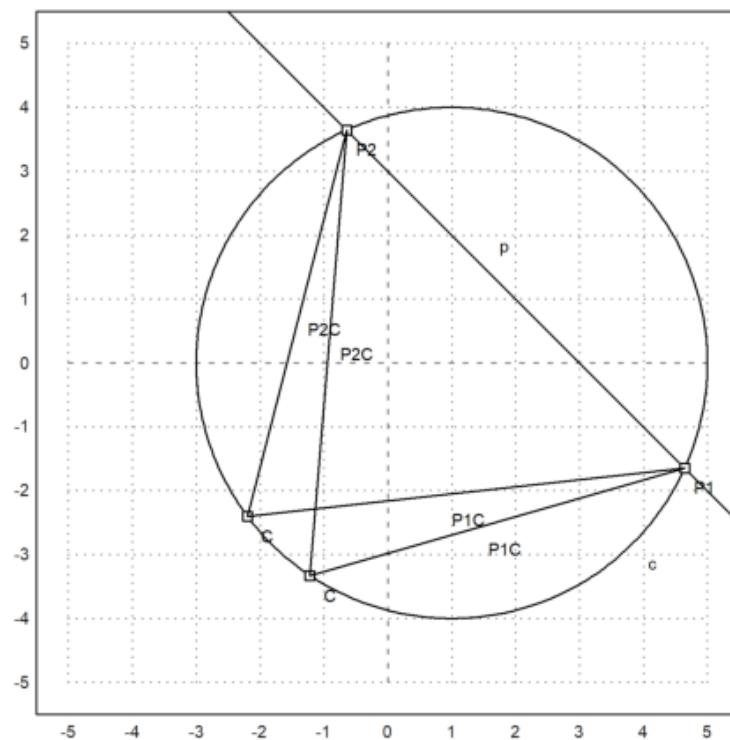
```
>C=A+normalize([-2,-3])*4; plotPoint(C); plotSegment(P1,C); plotSegment(P2,C);
>degprint(computeAngle(P1,C,P2))
```

$69^\circ 17' 42.68''$

```
>C=A+normalize([-4,-3])*4; plotPoint(C); plotSegment(P1,C); plotSegment(P2,C);
>degprint(computeAngle(P1,C,P2))
```

$69^\circ 17' 42.68''$

```
>insimg;
```



Garis Sumbu

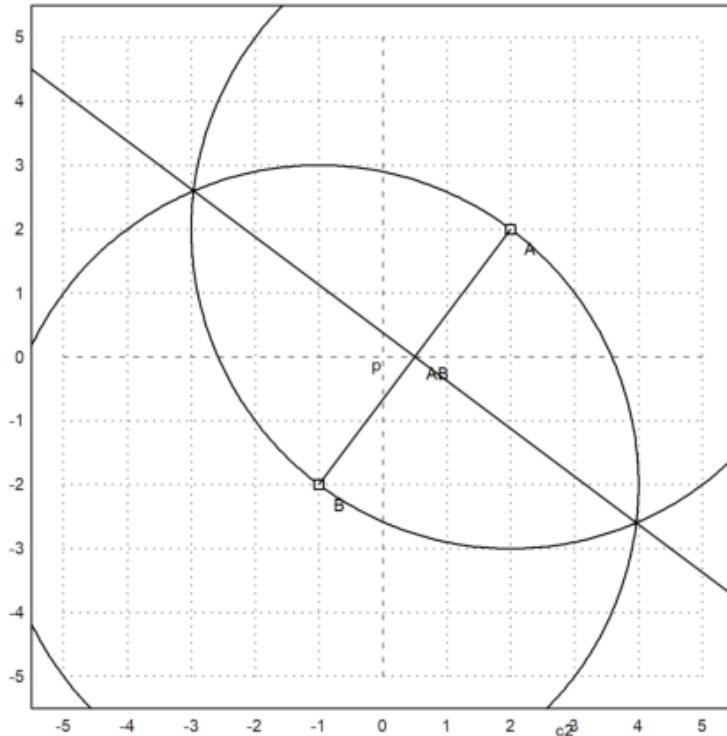
Berikut adalah langkah-langkah menggambar garis sumbu ruas garis AB:

1. Gambar lingkaran dengan pusat A melalui B.
2. Gambar lingkaran dengan pusat B melalui A.
3. Tarik garis melalui kedua titik potong kedua lingkaran tersebut. Garis ini merupakan garis sumbu (melalui titik tengah dan tegak lurus) AB.

```

>A=[2,2]; B=[-1,-2];
>c1=circleWithCenter(A,distance(A,B));
>c2=circleWithCenter(B,distance(A,B));
>{P1,P2,f}=circleCircleIntersections(c1,c2);
>l=lineThrough(P1,P2);
>setPlotRange(5); plotCircle(c1); plotCircle(c2);
>plotPoint(A); plotPoint(B); plotSegment(A,B); plotLine(l);

```



Selanjutnya kita melakukan hal yang sama di Maxima dengan koordinat umum.

```
>A &= [a1,a2]; B &= [b1,b2];
>c1 &= circleWithCenter(A,distance(A,B));
>c2 &= circleWithCenter(B,distance(A,B));
>P &= circleCircleIntersections(c1,c2); P1 &= P[1]; P2 &= P[2];
```

Persamaan untuk persimpangan cukup rumit. Tapi kita bisa menyederhanakannya jika kita mencari y.

```
>g &= getLineEquation(lineThrough(P1,P2),x,y);
>$solve(g,y)
```

$$\left[y = \frac{-(2b_1 - 2a_1)x + b_2^2 + b_1^2 - a_2^2 - a_1^2}{2b_2 - 2a_2} \right]$$

Ini memang sama dengan garis tengah tegak lurus, yang dihitung dengan cara yang sangat berbeda.

```
>$solve(getLineEquation(middlePerpendicular(A,B),x,y),y)
```

$$\left[y = \frac{-(2b_1 - 2a_1)x + b_2^2 + b_1^2 - a_2^2 - a_1^2}{2b_2 - 2a_2} \right]$$

```
>h &=getLineEquation(lineThrough(A,B),x,y);
>$solve(h,y)
```

$$\left[y = \frac{(b_2 - a_2)x - a_1 b_2 + a_2 b_1}{b_1 - a_1} \right]$$

Perhatikan hasil kali gradien garis g dan h adalah:

$$\frac{-(b_1 - a_1)}{(b_2 - a_2)} \times \frac{(b_2 - a_2)}{(b_1 - a_1)} = -1.$$

Artinya kedua garis tegak lurus.

Contoh 3: Rumus Heron

Rumus Heron menyatakan bahwa luas segitiga dengan panjang sisi-sisi a, b dan c adalah:

$$L = \sqrt{s(s-a)(s-b)(s-c)} \quad \text{dengan } s = (a+b+c)/2,$$

atau bisa ditulis dalam bentuk lain:

$$L = \frac{1}{4} \sqrt{(a+b+c)(b+c-a)(a+c-b)(a+b-c)}$$

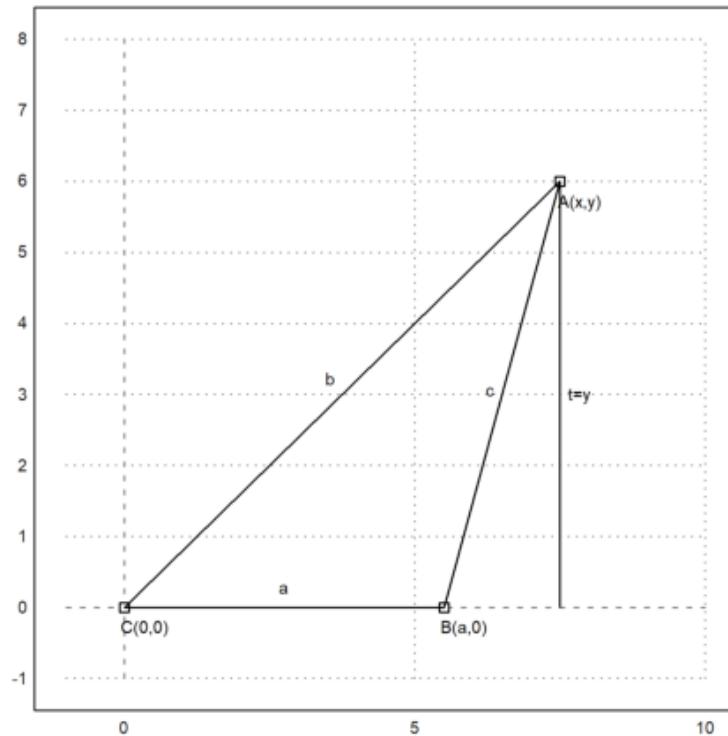
Untuk membuktikan hal ini kita misalkan C(0,0), B(a,0) dan A(x,y), b=AC, c=AB. Luas segitiga ABC adalah

$$L_{\triangle ABC} = \frac{1}{2}a \times y.$$

Nilai y didapat dengan menyelesaikan sistem persamaan:

$$x^2 + y^2 = b^2, \quad (x-a)^2 + y^2 = c^2.$$

```
>setPlotRange(-1,10,-1,8); plotPoint([0,0], "C(0,0)");
>plotPoint([7.5,6], "A(x,y)");
>plotSegment([0,0],[5.5,0], "a",25); plotSegment([5.5,0],[7.5,6],"c",15);
>plotSegment([0,0],[7.5,6],"b",25);
>plotSegment([7.5,6],[7.5,0],"t=y",25);
```



```
>&assume(a>0); sol &= solve([x^2+y^2=b^2, (x-a)^2+y^2=c^2], [x, y])
```

[]

Ekstrak solusi y.

```
>ysol &= y with sol[2][2]; $'y=sqrt(factor(ysol^2))
```

Maxima said:

```
part: invalid index of list or matrix.  
-- an error. To debug this try: debugmode(true);
```

Error in:

```
ysol &= y with sol[2][2]; $'y=sqrt(factor(ysol^2)) ...  
^
```

Kita mendapatkan rumus Heron.

```
>function H(a,b,c) &= sqrt(factor((ysol*a/2)^2)); $'H(a,b,c)=H(a,b,c)
```

$$H(a, b, [1, 0, 4]) = \frac{a |ysol|}{2}$$

```
>$' Luas=H(2,5,6) // luas segitiga dengan panjang sisi-sisi 2, 5, 6
```

$$Luas = |ysol|$$

Tentu saja, setiap segitiga siku-siku adalah masalah yang diketahui.

```
>H(3,4,5) //luas segitiga siku-siku dengan panjang sisi 3, 4, 5
```

Variable or function ysol not found.

Try "trace errors" to inspect local variables after errors.

H:

```
useglobal; return a*abs(ysol)/2
```

Error in:

```
H(3,4,5) //luas segitiga siku-siku dengan panjang sisi 3, 4, 5 ...
^
```

Dan jelas juga bahwa ini adalah segitiga dengan luas maksimal dan kedua sisinya 3 dan 4.

```
>aspect (1.5); plot2d(&H(3,4,x),1,7); // Kurva luas segitiga sengan panjang
```

Variable or function ysol not found.

Error in expression: 3*abs(ysol)/2

%ploteval:

```
y0=f$(x[1],args());
```

adaptiveevalone:

```
s=%ploteval(g$,t,args());
```

Try "trace errors" to inspect local variables after errors.

plot2d:

```
dw/n,dw/n^2,dw/n,auto,args());
```

Kasus umum juga berhasil.

```
>$solve(diff(H(a,b,c)^2,c)=0,c)
```

Maxima said:

```
diff: second argument must be a variable; found [1,0,4]
-- an error. To debug this try: debugmode(true);
```

```
Error in:
\$solve(diff(H(a,b,c)^2,c)=0,c) ...
^
```

Sekarang mari kita cari himpunan semua titik di mana $b+c=d$ untuk suatu konstanta d . Diketahui bahwa ini adalah ellips.

```
>s1 &= subst(d-c,b,sol[2]); $s1
```

```
Maxima said:
part: invalid index of list or matrix.
-- an error. To debug this try: debugmode(true);
```

```
Error in:
s1 &= subst(d-c,b,sol[2]); $s1 ...
^
```

Dan buatlah fungsinya.

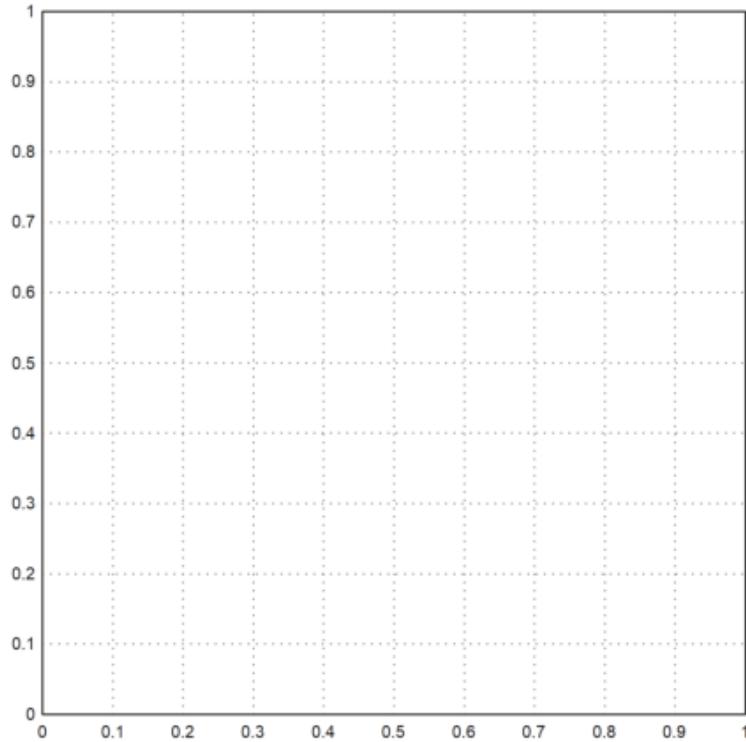
```
>function fx(a,c,d) &= rhs(s1[1]); $fx(a,c,d), function fy(a,c,d) &= rhs(s1
```

0

0

Sekarang kita bisa menggambar setnya. Sisi b bervariasi dari 1 sampai 4. Diketahui bahwa kita memperoleh ellips.

```
>aspect(1); plot2d(&fx(3,x,5),&fy(3,x,5),xmin=1,xmax=4,square=1):
```



Kita dapat memeriksa persamaan umum elips ini, yaitu:

$$\frac{(x - x_m)^2}{u^2} + \frac{(y - y_m)^2}{v^2} = 1,$$

dimana (x_m, y_m) adalah pusat, dan u dan v adalah setengah sumbu.

```
>ratsimp((fx(a,c,d)-a/2)^2/u^2+fy(a,c,d)^2/v^2 with [u=d/2,v=sqrt(d^2-a^2)]
```

$$\frac{a^2}{d^2}$$

Kita melihat bahwa tinggi dan luas segitiga adalah maksimal untuk $x=0$. Jadi luas segitiga dengan $a+b+c=d$ adalah maksimal jika segitiga tersebut sama sisi. Kami ingin memperolehnya secara analitis.

```
>eqns &= [diff(H(a,b,d-(a+b))^2,a)=0,diff(H(a,b,d-(a+b))^2,b)=0]; $eqns
```

$$\left[\frac{a y_{sol}^2}{2} = 0, 0 = 0 \right]$$

Kita mendapatkan nilai minimum yang dimiliki oleh segitiga dengan salah satu sisinya 0, dan solusinya $a=b=c=d/3$.

```
>$solve(eqns, [a,b])
```

$$[[a = 0, b = \%r_1]]$$

Ada juga metode Lagrange, yang memaksimalkan $H(a,b,c)^2$ terhadap $a+b+c=d$.

```
>&solve([diff(H(a,b,c)^2,a)=la,diff(H(a,b,c)^2,b)=la, ...
>      diff(H(a,b,c)^2,c)=la,a+b+c=d],[a,b,c,la])
```

Maxima said:

```
diff: second argument must be a variable; found [1,0,4]
-- an error. To debug this try: debugmode(true);
```

Error in:

```
... la,      diff(H(a,b,c)^2,c)=la,a+b+c=d],[a,b,c,la]) ...
^
```

Kita dapat membuat plot dari situasi tersebut.

Pertama atur titik di Maxima.

```
>A &= at([x,y],sol[2]); $A
```

Maxima said:

```
part: invalid index of list or matrix.
-- an error. To debug this try: debugmode(true);
```

Error in:

```
A &= at([x,y],sol[2]); $A ...
^
```

```
>B &= [0,0]; $B, C &= [a,0]; $C
```

$$[0, 0]$$

$$[a, 0]$$

Kemudian atur rentang plot, dan plot titik-titiknya.

```
>setPlotRange(0,5,-2,3); ...
>a=4; b=3; c=2; ...
>plotPoint(mxmeval("B"), "B"); plotPoint(mxmeval("C"), "C"); ...
>plotPoint(mxmeval("A"), "A");
```

Variable a1 not found!
 Use global variables or parameters for string evaluation.
 Error in Evaluate, superfluous characters found.
 Try "trace errors" to inspect local variables after errors.
 mxmeval:
 return evaluate(mxm(s));
 Error in:
 ... otPoint(mxmeval("C"), "C"); plotPoint(mxmeval("A"), "A"): ...
 ^

Plot segmennya.

```
>plotSegment(mxmeval("A"),mxmeval("C")); ...
>plotSegment(mxmeval("B"),mxmeval("C")); ...
>plotSegment(mxmeval("B"),mxmeval("A"));
```

Variable a1 not found!
 Use global variables or parameters for string evaluation.
 Error in Evaluate, superfluous characters found.
 Try "trace errors" to inspect local variables after errors.
 mxmeval:
 return evaluate(mxm(s));
 Error in:
 plotSegment(mxmeval("A"),mxmeval("C")); plotSegment(mxmeval("B") ...
 ^

Hitung garis tengah tegak lurus di Maxima.

```
>h &= middlePerpendicular(A,B); g &= middlePerpendicular(B,C);
```

Dan pusat lingkarannya.

```
>U &= lineIntersection(h,g);
```

Kita mendapatkan rumus jari-jari lingkaran luar.

```
>&assume (a>0,b>0,c>0); $distance(U,B) | radcan
```

$$\frac{\sqrt{a_2^2 + a_1^2} \sqrt{a_2^2 + a_1^2 - 2 a a_1 + a^2}}{2 |a_2|}$$

Mari kita tambahkan ini ke dalam plot.

```
>plotPoint(U()); ...
>plotCircle(circleWithCenter(mxmeval("U"),mxmeval("distance(U,C)"))):
```

```
Variable a2 not found!
Use global variables or parameters for string evaluation.
Error in ^
Error in expression: [a/2, (a2^2+a1^2-a*a1)/(2*a2)]
Error in:
plotPoint(U()); plotCircle(circleWithCenter(mxmeval("U"),mxmev ...
^
```

Dengan menggunakan geometri, kita memperoleh rumus sederhana

$$\frac{a}{\sin(\alpha)} = 2r$$

untuk radius. Kita bisa cek, apakah ini benar adanya pada Maxima. Maxima akan memfaktorkan ini hanya jika kita mengkuadratkannya.

```
>$c^2/sin(computeAngle(A,B,C))^2 | factor
```

$$\left[\frac{a_2^2 + a_1^2}{a_2^2}, 0, \frac{16 (a_2^2 + a_1^2)}{a_2^2} \right]$$

Contoh 4: Garis Euler dan Parabola

Garis Euler adalah garis yang ditentukan dari sembarang segitiga yang tidak sama sisi. Merupakan garis tengah segitiga, dan melewati beberapa titik penting yang ditentukan dari segitiga, antara lain ortocenter, sirkumcenter, centroid, titik Exeter dan pusat lingkaran sembilan titik segitiga.

Untuk demonstrasinya, kita menghitung dan memplot garis Euler dalam sebuah segitiga. Pertama, kita mendefinisikan sudut-sudut segitiga di Euler. Kami menggunakan definisi, yang terlihat dalam ekspresi simbolik.

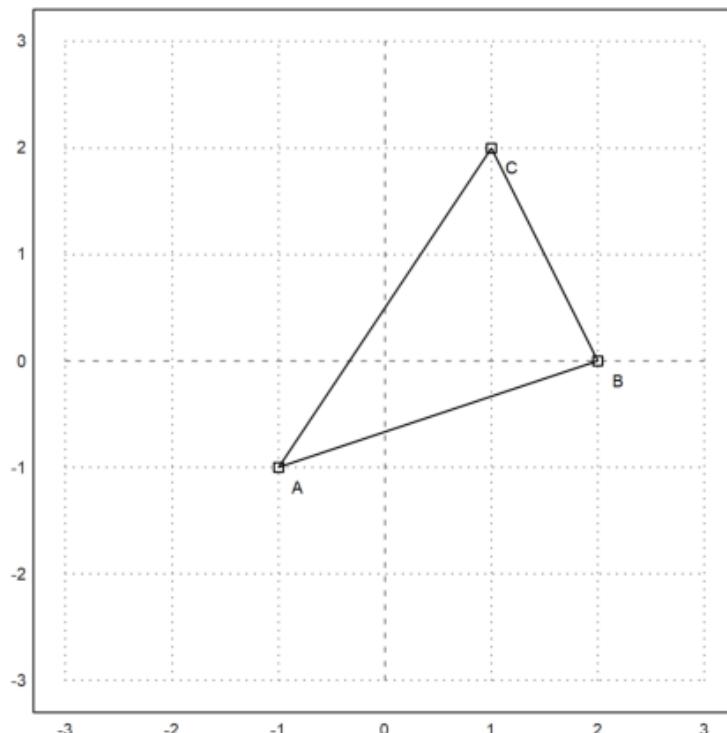
```
>A:::=[-1,-1]; B:::[2,0]; C:::[1,2];
```

Untuk memplot objek geometris, kita menyiapkan area plot, dan menambahkan titik ke dalamnya. Semua plot objek geometris ditambahkan ke plot saat ini.

```
>setPlotRange(3); plotPoint(A, "A"); plotPoint(B, "B"); plotPoint(C, "C");
```

Kita juga bisa menjumlahkan sisi-sisi segitiga.

```
>plotSegment(A,B,""); plotSegment(B,C,""); plotSegment(C,A,"");
```



Berikut luas segitiga menggunakan rumus determinan. Tentu saja kami harus mengambil nilai absolut dari hasil ini.

```
>$areaTriangle(A,B,C)
```

$$-\frac{7}{2}$$

Kita dapat menghitung koefisien sisi c.

```
>c &= lineThrough(A,B)
```

$$[-1, 3, -2]$$

Dan dapatkan juga rumus untuk baris ini.

```
>$getLineEquation(c,x,y)
```

$$3y - x = -2$$

Untuk bentuk Hesse, kita perlu menentukan sebuah titik, sehingga titik tersebut berada di sisi positif dari Hesseform. Memasukkan titik akan menghasilkan jarak positif ke garis.

```
>$getHesseForm(c,x,y,C), $at(%,[x=C[1],y=C[2]])
```

$$\frac{3y - x + 2}{\sqrt{10}}$$

$$\frac{7}{\sqrt{10}}$$

Sekarang kita menghitung lingkaran luar ABC.

```
>LL &= circleThrough(A,B,C); $getCircleEquation(LL,x,y)
```

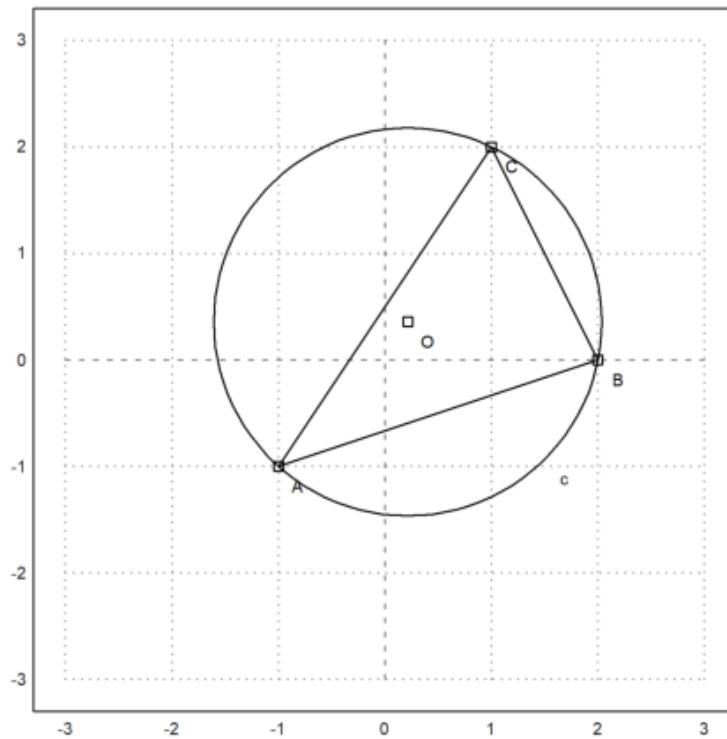
$$\left(y - \frac{5}{14}\right)^2 + \left(x - \frac{3}{14}\right)^2 = \frac{325}{98}$$

```
>O &= getCircleCenter(LL); $O
```

$$\left[\frac{3}{14}, \frac{5}{14}\right]$$

Plot lingkaran dan pusatnya. Cu dan U bersifat simbolis. Kami mengevaluasi ekspresi ini untuk Euler.

```
>plotCircle(LL()); plotPoint(O(), "O"):
```



Kita dapat menghitung perpotongan ketinggian di ABC (ortocenter) secara numerik dengan perintah berikut.

```
>H &= lineIntersection(perpendicular(A, lineThrough(C, B)), ...
>    perpendicular(B, lineThrough(A, C))); $H
```

$$\left[\frac{11}{7}, \frac{2}{7} \right]$$

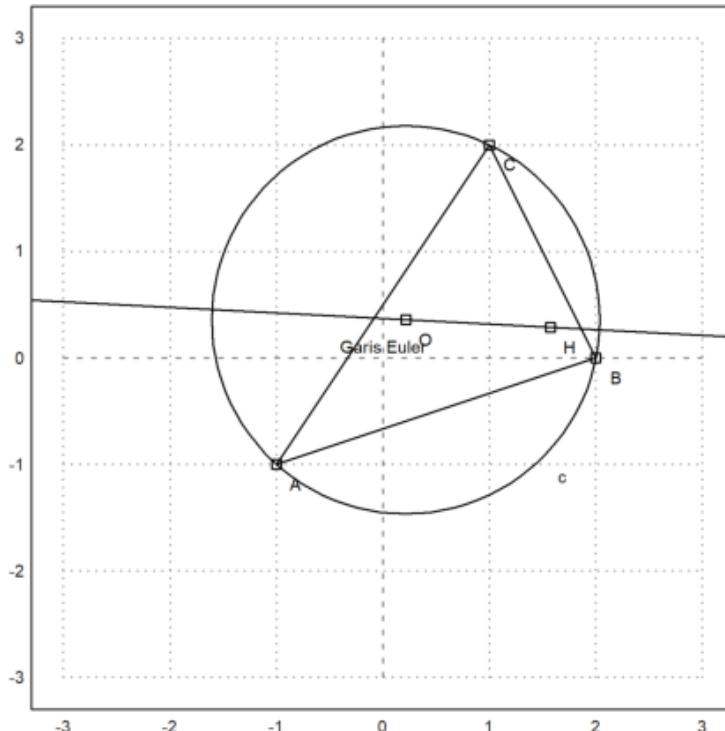
Sekarang kita dapat menghitung garis segitiga Euler.

```
>el &= lineThrough(H, O); $getLineEquation(el, x, y)
```

$$-\frac{19y}{14} - \frac{x}{14} = -\frac{1}{2}$$

Tambahkan ke plot kita.

```
>plotPoint(H(),"H"); plotLine(el(),"Garis Euler");
```

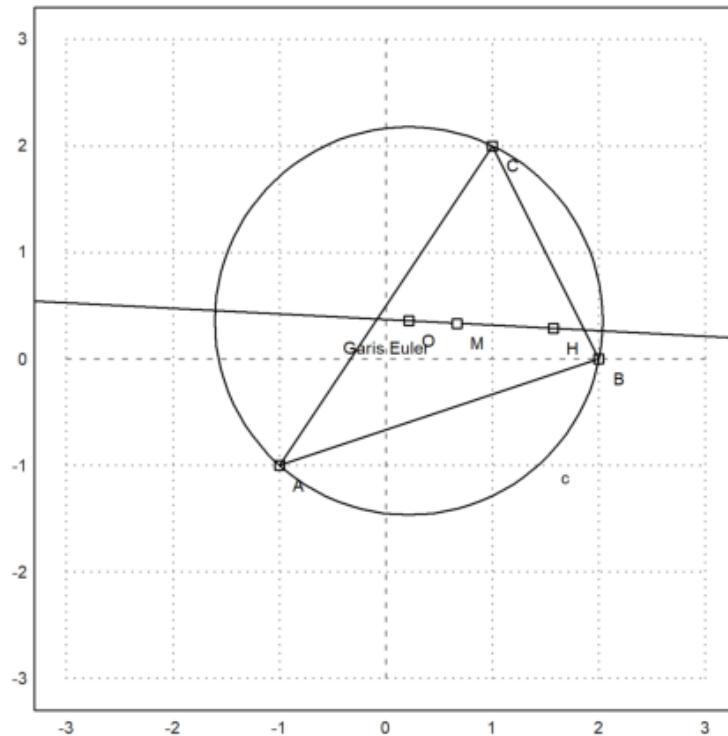


Pusat gravitasi seharusnya berada di garis ini.

```
>M &= (A+B+C)/3; $getLineEquation(el,x,y) with [x=M[1],y=M[2]]
```

$$-\frac{1}{2} = -\frac{1}{2}$$

```
>plotPoint(M(),"M"); // titik berat
```



Teorinya memberitahu kita $MH=2*MO$. Kita perlu menyederhanakan dengan radcan untuk mencapai hal ini.

```
>$distance(M, H) / distance(M, O) | radcan
```

2

Fungsinya mencakup fungsi untuk sudut juga.

```
>$computeAngle(A, C, B), degprint(%())
```

$$\arccos\left(\frac{4}{\sqrt{5}\sqrt{13}}\right)$$

$60^\circ 15' 18.43''$

Persamaan pusat lingkaran tidak terlalu bagus.

```
>Q &= lineIntersection(angleBisector(A, C, B), angleBisector(C, B, A)) | radcan; $
```

$$\left[\frac{\left(2^{\frac{3}{2}} + 1\right) \sqrt{5} \sqrt{13} - 15 \sqrt{2} + 3}{14}, \frac{(\sqrt{2} - 3) \sqrt{5} \sqrt{13} + 5 2^{\frac{3}{2}} + 5}{14} \right]$$

Mari kita hitung juga ekspresi jari-jari lingkaran yang tertulis.

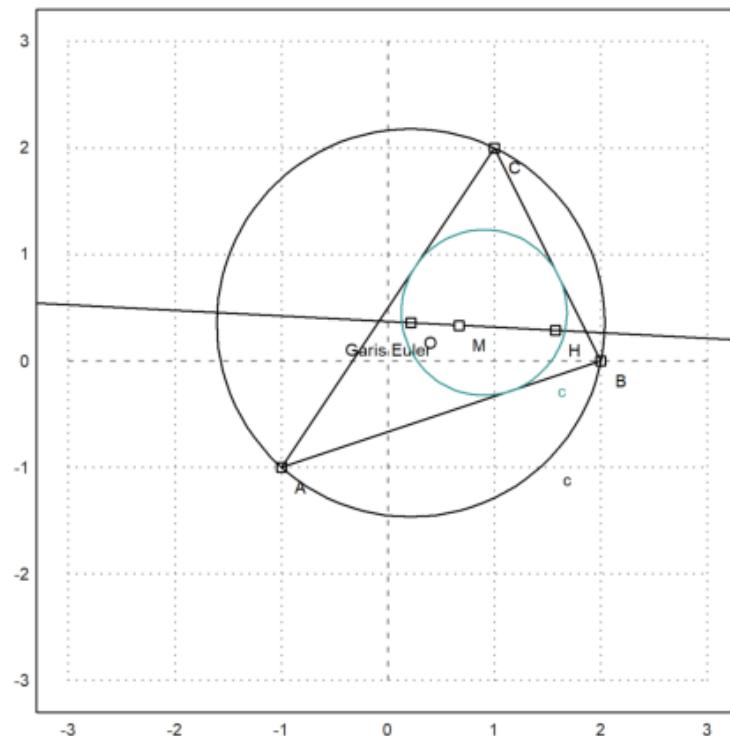
```
>r &= distance(Q,projectToLine(Q,lineThrough(A,B))) | ratsimp; $r
```

$$\frac{\sqrt{(-41\sqrt{2} - 31)\sqrt{5}\sqrt{13} + 115\sqrt{2} + 614}}{7\sqrt{2}}$$

```
>LD &= circleWithCenter(Q,r); // Lingkaran dalam
```

Mari kita tambahkan ini ke dalam plot.

```
>color(5); plotCircle(LD());
```



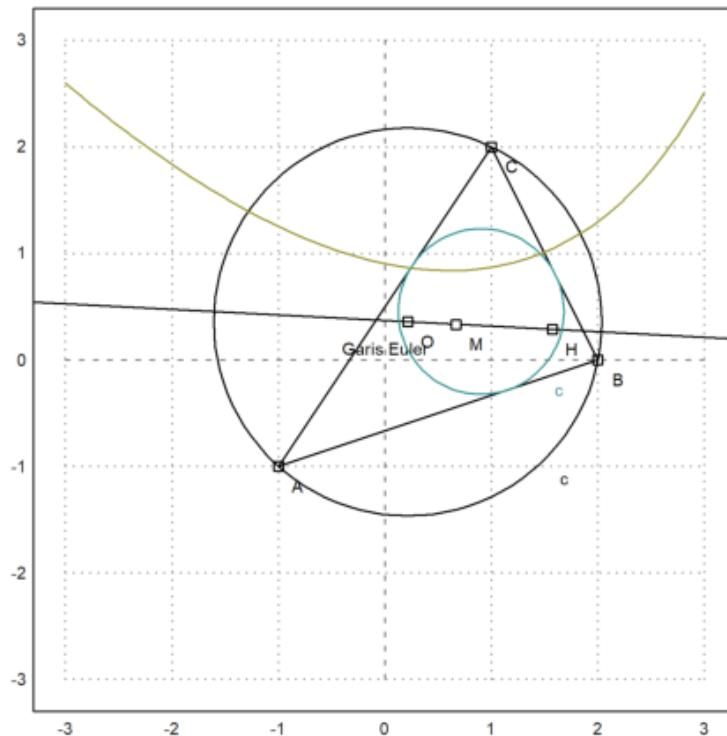
Selanjutnya akan dicari persamaan tempat kedudukan titik-titik yang berjarak sama ke titik C dan ke garis AB.

```
>p &= getHesseForm(lineThrough(A,B),x,y,C)-distance([x,y],C); $p='0
```

$$\frac{3y - x + 2}{\sqrt{10}} - \sqrt{(2-y)^2 + (1-x)^2} = 0$$

Persamaan tersebut dapat digambar menjadi satu dengan gambar sebelumnya.

```
>plot2d(p,level=0,add=1,contourcolor=6):
```



This should be some function, but the default solver of Maxima can find the solution only, if we square the equation. Consequently, we get a fake solution.

```
>akar &= solve(getHesseForm(lineThrough(A,B),x,y,C)^2-distance([x,y],C)^2,y)
```

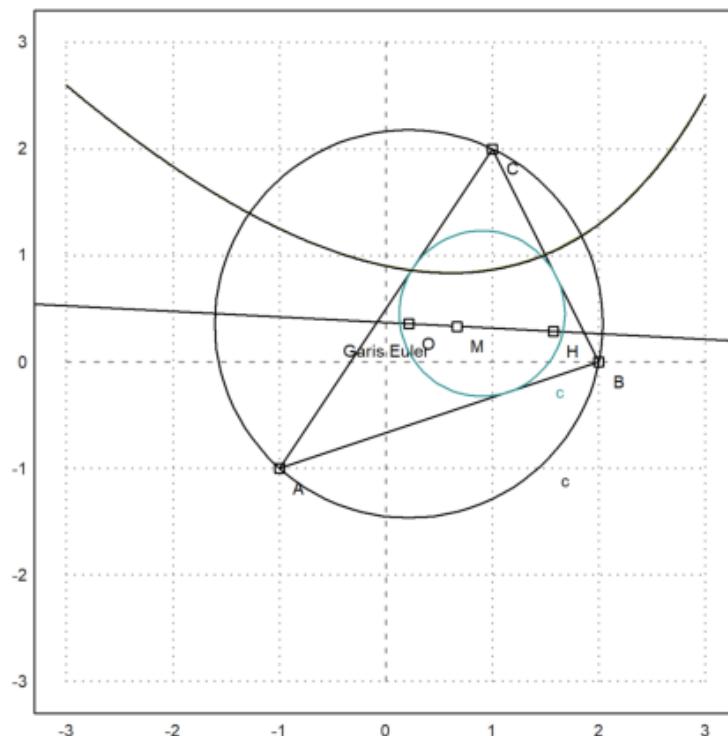
$$\begin{aligned} y &= -3x - \sqrt{70} \sqrt{9 - 2x} + 26, \\ y &= -3x + \sqrt{70} \sqrt{9 - 2x} + 26 \end{aligned}$$

The first solution is

maxima: akar[1]

Adding the first solution to the plot show, that it is indeed the path we are looking for. The theory tells us that it is a rotated parabola.

```
>plot2d(&rhs(akar[1]), add=1):
```



```
>function g(x) &= rhs(akar[1]); $'g(x)= g(x) // fungsi yang mendefinisikan k
```

$$g(x) = -3x - \sqrt{70} \sqrt{9 - 2x} + 26$$

```
>T &=[-1, g(-1)]; // ambil sebarang titik pada kurva tersebut
>dTC &= distance(T,C); $fullratsimp(dTC), $float(%); // jarak T ke C
```

$$\sqrt{1503 - 54\sqrt{11}\sqrt{70}}$$

$$2.135605779339061$$

```
>U &= projectToLine(T,lineThrough(A,B)); $U // proyeksi T pada garis AB
```

$$\left[\frac{80 - 3\sqrt{11}\sqrt{70}}{10}, \frac{20 - \sqrt{11}\sqrt{70}}{10} \right]$$

```
>dU2AB &= distance(T,U); $fullratsimp(dU2AB), $float(%) // jarak T ke AB
```

$$\sqrt{1503 - 54\sqrt{11}\sqrt{70}}$$

$$2.135605779339061$$

Ternyata jarak T ke C sama dengan jarak T ke AB. Coba Anda pilih titik T yang lain dan ulangi perhitungan-perhitungan di atas untuk menunjukkan bahwa hasilnya juga sama.

Contoh 5: Trigonometri Rasional

This is inspired by a talk N.J.Wildberger. In his book "Divine Proportions", Wildberger proposes to replace the classical notions of distance and angle by quadrance and spread. Using these, it is indeed possible to avoid trigonometric functions in many examples, and to stay "rational".

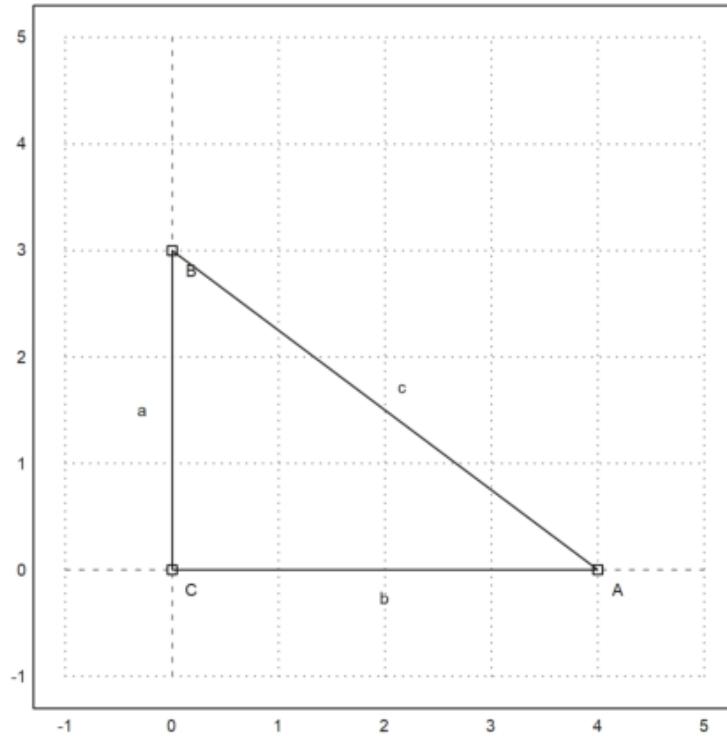
In the following, I introduce the concepts, and solve a few problems. I am using symbolic computations Maxima here, which hides the main advantage of rational trigonometry that the computations can be performed by paper and pencil only. You are invited to check the results without a computer.

The point is that symbolic rational computations often yield simple results. In contrast, classical trigonometry yields complicated trigonometric results, which evaluate to numerical approximations only.

```
>load geometry;
```

For a first introduction, we use the rectangular triangle with the famous Egyptian proportions 3, 4 and 5. The following commands are Euler commands to plot plane geometry contained in the Euler file "geometry.e".

```
>C&:=[0,0]; A&:=[4,0]; B&:=[0,3]; ...
>setPlotRange(-1,5,-1,5); ...
>plotPoint(A,"A"); plotPoint(B,"B"); plotPoint(C,"C"); ...
>plotSegment(B,A,"c"); plotSegment(A,C,"b"); plotSegment(C,B,"a"); ...
>insimg(30);
```



Of course,

$$\sin(w_a) = \frac{a}{c},$$

where w_a is the angle at A. The usual way to compute this angle, is to take the inverse of the sine function. The result is an undigestible angle, which can only be printed approximately.

```
>wa := arcsin(3/5); degprint(wa)
```

$36^\circ 52' 11.63''$

Rational trigonometry tries to avoid this.

The first notion of rational trigonometry is a quadrance, which replaces the distance. In fact, it is just the distance squared. In the following, a, b, and c denote the quadrances of the sides.

The Pythagoras theorem simply becomes $a+b=c$ then.

```
>a &= 3^2; b &= 4^2; c &= 5^2; &a+b=c
```

$$25 = 25$$

The second notion of rational trigonometry is the spread. The spread measures the opening between lines. It is 0, if the lines are parallel, and 1, if the lines are rectangular. It is the square of the sine of the angle between the two lines.

The spread of the lines AB and AC in the image above is defined as

$$s_a = \sin(\alpha)^2 = \frac{a}{c},$$

where a and c are the quadrances of any rectangular triangle with one corner in A.

```
>sa &= a/c; $sa
```

$$\frac{9}{25}$$

This is easier to compute than the angle, of course. But you lose the property that angles can be added easily.

Of course, we can convert our approximate value for the angle wa to a sprad, and print it as a fraction.

```
>fracprint(sin(wa)^2)
```

$$9/25$$

The cosine law of classical trigonometry translates into the following "cross law".

$$(c + b - a)^2 = 4bc(1 - s_a)$$

Here a, b, and c are quadrances of the sides of a triangle, and sa is the spread of at the corner A. The side a is, as usual, opposite to the corner A.

These laws are implemented in the geometry.e file we loaded into Euler.

```
>$crosslaw(aa,bb,cc,saa)
```

$$\left[\left(bb - aa + \frac{7}{6} \right)^2, \left(bb - aa + \frac{7}{6} \right)^2, \left(bb - aa + \frac{5}{3\sqrt{2}} \right)^2 \right] = \left[\frac{14bb(1-saa)}{3}, \frac{14bb(1-saa)}{3}, \frac{52^{\frac{3}{2}}bb(1-saa)}{3} \right]$$

In our case we get

```
>$crosslaw(a,b,c,sa)
```

$$1024 = 1024$$

Let us use this crosslaw to find the spread at A. To do this, we produce the crosslaw for the quadrances a , b , and c , and solve it for the unknown spread sa .

You can do this by hand easily, but I use Maxima. Of course, we get the result, we already had.

```
>$crosslaw(a,b,c,x), $solve(%,x)
```

$$1024 = 1600 (1 - x)$$

$$\left[x = \frac{9}{25} \right]$$

We know this already. The definition of the spread is a special case of the crosslaw.

We can also solve this for general a, b, c . The result is a formula which computes the spread of an angle of a triangle given the quadrances of the three sides.

```
>$solve(crosslaw(aa,bb,cc,x),x)
```

$$\left[\left[\frac{168 bb x + 36 bb^2 + (-72 aa - 84) bb + 36 aa^2 - 84 aa + 49}{36}, \frac{168 bb x + 36 bb^2 + (-72 aa - 84) bb + 36 aa^2 - 84 aa + 49}{36} \right] \right]$$

We could make a function of the result. Such a function is already defined in the geometry.e file of Euler.

```
>$spread(a,b,c)
```

$$\frac{9}{25}$$

As an example, we can use it to compute the angle of a triangle with sides

$$a, \quad a, \quad \frac{4a}{7}$$

The result is rational, which is not so easy to get if we use classical trigonometry.

```
>$spread(a,a,4*a/7)
```

$$\frac{6}{7}$$

This is the angle in degrees.

```
>degprint(arcsin(sqrt(6/7)))
```

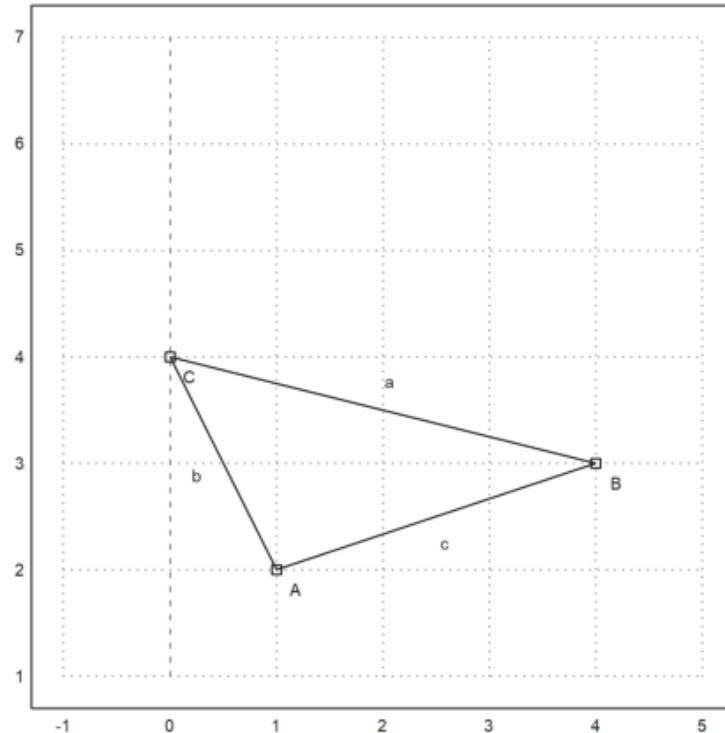
$67^\circ 47' 32.44''$

Another Example

Now, let us try a more advanced example.

We set three corners of a triangle as follows.

```
>A&:=[1,2]; B&:=[4,3]; C&:=[0,4]; ...
>setPlotRange (-1,5,1,7); ...
>plotPoint (A, "A"); plotPoint (B, "B"); plotPoint (C, "C"); ...
>plotSegment (B,A,"c"); plotSegment (A,C,"b"); plotSegment (C,B,"a"); ...
>insimg;
```



Using Pythagoras, it is easy to compute the distance between two points. I first use the function `distance` of the Euler file for geometry. The function `distance` uses classical geometry.

```
>$distance (A, B)
```

$$\sqrt{10}$$

Euler does also contain functions for the quadrance between two points.

In the following example, since $c+b$ is not a , the triangle is not rectangular.

```
>c &= quad(A, B); $c, b &= quad(A, C); $b, a &= quad(B, C); $a,
```

$$10$$

$$5$$

$$17$$

First, let us compute the traditional angle. The function `computeAngle` uses the usual method based on the dot product of two vectors. The result is some floating point approximation.

$$A = \langle 1, 2 \rangle \quad B = \langle 4, 3 \rangle, \quad C = \langle 0, 4 \rangle$$

$$\mathbf{a} = C - B = \langle -4, 1 \rangle, \quad \mathbf{c} = A - B = \langle -3, -1 \rangle, \quad \beta = \angle ABC$$

$$\mathbf{a} \cdot \mathbf{c} = |\mathbf{a}| \cdot |\mathbf{c}| \cos \beta$$

$$\cos \angle ABC = \cos \beta = \frac{\mathbf{a} \cdot \mathbf{c}}{|\mathbf{a}| \cdot |\mathbf{c}|} = \frac{12 - 1}{\sqrt{17} \sqrt{10}} = \frac{11}{\sqrt{17} \sqrt{10}}$$

```
>wb &= computeAngle(A, B, C); $wb, $(wb/pi*180)()
```

$$\arccos \left(\frac{11}{\sqrt{10} \sqrt{17}} \right)$$

$$32.4711922908$$

Using pencil and paper, we can do the same with the cross law. We insert the quadrances a , b , and c into the cross law and solve for x .

```
>$crosslaw(a,b,c,x), $solve(%,x), // (b+c-a)^=4b.c(1-x)
```

$$4 = 200 (1 - x)$$

$$\left[x = \frac{49}{50} \right]$$

That is, what the function spread defined in "geometry.e" does.

```
>sb &= spread(b,a,c); $sb
```

$$\frac{49}{170}$$

Maxima gets the same result using ordinary trigonometry, if we force it. It does resolve the $\sin(\arccos(...))$ term to a fractional result. Most students could not do this.

```
>$sin(computeAngle(A,B,C))^2
```

$$\frac{49}{170}$$

Once, we have the spread at B, we can compute the height ha on the side a . Remember that

$$s_b = \frac{h_a}{c}$$

by definition.

```
>ha &= c*sb; $ha
```

$$\frac{49}{17}$$

The following image has been produced with the geometry program C.a.R., which can draw quadrances and spreads.

image: (20) Rational_Geometry_CaR.png

By definition the length of ha is the square root of its quadrance.

```
>$sqrt(ha)
```

$$\frac{7}{\sqrt{17}}$$

Now we can compute the area of the triangle. Do not forget, that we are dealing with quadrances!

```
>$sqrt(ha)*sqrt(a)/2
```

$$\frac{7}{2}$$

The usual determinant formula yields the same result.

```
>$areaTriangle(B,A,C)
```

$$\frac{7}{2}$$

The Heron Formula

Now, let us solve this problem in general!

```
>&remvalue(a,b,c,sb,ha);
```

We first compute the spread at B for a triangle with sides a , b , and c . Then we compute the area squared (the "quadrea"?), factor it with Maxima, and we get the famous formula of Heron.

Admittedly, this is tough to do with pencil and paper.

```
>$spread(b^2,c^2,a^2), $factor(%*c^2*a^2/4)
```

$$\frac{-c^4 - (-2b^2 - 2a^2)c^2 - b^4 + 2a^2b^2 - a^4}{4a^2c^2}$$

$$\frac{(-c + b + a)(c - b + a)(c + b - a)(c + b + a)}{16}$$

The Triple Spread Rule

The disadvantage of spreads is that they do no longer simply add like angles.

However, the three spreads of a triangle satisfy the following "triple spread" rule.

```
>&remvalue(sa, sb, sc); $triplespread(sa, sb, sc)
```

$$(sc + sb + sa)^2 = 2(sc^2 + sb^2 + sa^2) + 4sa sb sc$$

This rule is valid for any three angles that add to 180° .

$$\alpha + \beta + \gamma = \pi$$

Since the spreads of

$$\alpha, \pi - \alpha$$

are equal, the triple spread rule is also true, if

$$\alpha + \beta = \gamma$$

Since the spread of the negative angle is the same, the triple spread rule also holds, if

$$\alpha + \beta + \gamma = 0$$

For example, we can compute the spread of the 60° angle. It is $3/4$. The equations have a second solution, however, where all spreads are 0.

```
>$solve(triplespread(x, x, x), x)
```

$$\left[x = \frac{3}{4}, x = 0 \right]$$

The spread of 90° is obviously 1. If two angles add to 90° , their spread solves the triple spread equation with $a,b,1$. By the following computation we get $a+b=1$.

```
>$triplespread(x,y,1), $solve(%,x)
```

$$(y + x + 1)^2 = 2 (y^2 + x^2 + 1) + 4xy$$

$$[x = 1 - y]$$

Since the spread of $180^\circ - t$ is the same as the spread of t , the triple spread formula also holds, if one angle is the sum or difference of the two other angles.

So we can find the spread of the doubled angle. Note that there are two solutions again. We make this a function.

```
>$solve(triplespread(a,a,x),x), function doublespread(a) &= factor(rhs(%[1])
```

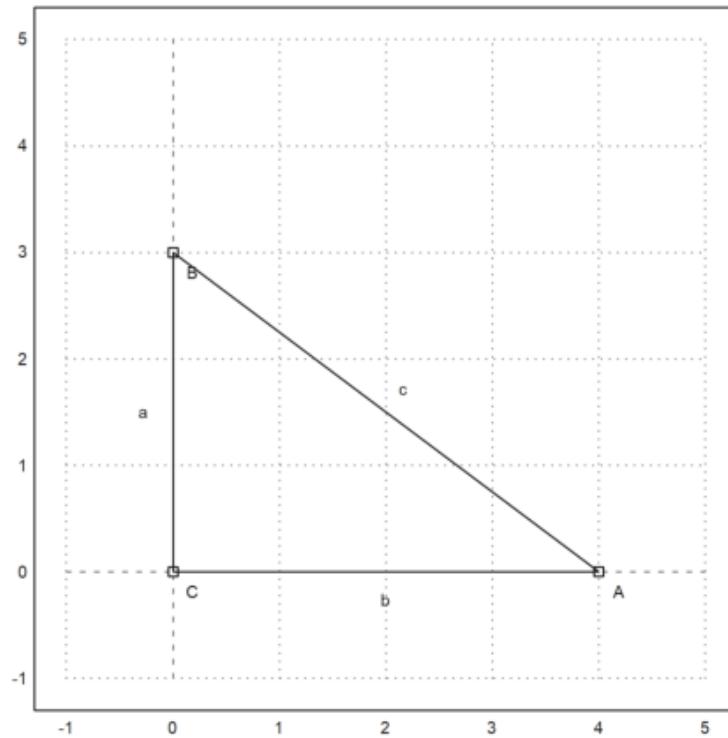
$$[x = 4a - 4a^2, x = 0]$$

$$- 4(a - 1)a$$

Angle Bisectors

This is the situation, we already know.

```
>C:=[0,0]; A:=[4,0]; B:=[0,3]; ...
>setPlotRange(-1,5,-1,5); ...
>plotPoint(A, "A"); plotPoint(B, "B"); plotPoint(C, "C"); ...
>plotSegment(B,A,"c"); plotSegment(A,C,"b"); plotSegment(C,B,"a"); ...
>insimg;
```



Let us compute the length of the angle bisector at A. But we want to solve that for general a,b,c.

```
>&remvalue(a,b,c);
```

So we first compute the spread of the bisected angle at A, using the triple spread formula.

The problem with this formula shows up again. It has two solutions. We have to pick the correct one. The other solution refers to the bisected angle 180° -wa.

```
>$triplespread(x,x,a/(a+b)), $solve(% ,x), sa2 &= rhs(%[1]); $sa2
```

$$\left(2x + \frac{a}{b+a}\right)^2 = 2 \left(2x^2 + \frac{a^2}{(b+a)^2}\right) + \frac{4ax^2}{b+a}$$

$$\left[x = \frac{-\sqrt{b}\sqrt{b+a} + b + a}{2b + 2a}, x = \frac{\sqrt{b}\sqrt{b+a} + b + a}{2b + 2a}\right]$$

$$\frac{-\sqrt{b}\sqrt{b+a} + b + a}{2b + 2a}$$

Let us check for the Egyptian rectangle.

```
>$sa2 with [a=3^2,b=4^2]
```

$$\frac{1}{10}$$

We can print the angle in Euler, after transferring the spread to radians.

```
>wa2 := arcsin(sqrt(1/10)); degprint(wa2)
```

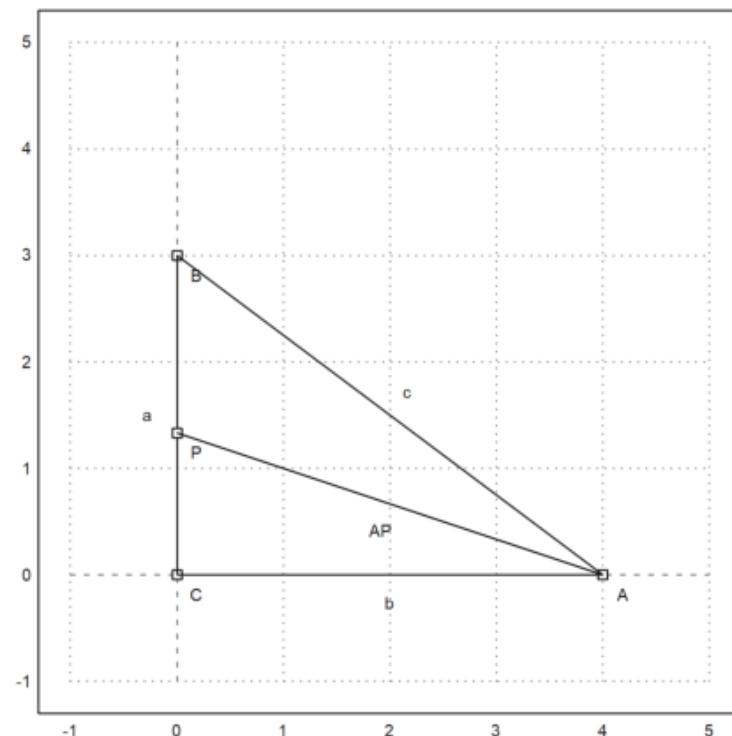
$$18^\circ 26' 5.82''$$

The point P is the intersection of the angle bisector with the y-axis.

```
>P := [0, tan(wa2)*4]
```

$$[0, 1.33333]$$

```
>plotPoint(P, "P"); plotSegment(A, P);
```



Let us check the angles in our specific example.

```
>computeAngle(C,A,P), computeAngle(P,A,B)
```

0.321750554397

0.321750554397

Now we compute the length of the bisector AP.

We use the sine theorem in the triangle APC. This theorem states that

$$\frac{BC}{\sin(w_a)} = \frac{AC}{\sin(w_b)} = \frac{AB}{\sin(w_c)}$$

holds in any triangle. Square it, it translates into the so-called "spread law"

$$\frac{a}{s_a} = \frac{b}{s_b} = \frac{c}{s_b}$$

where a,b,c denote quadrances.

Since the spread CPA is $1-sa^2$, we get from it $bisa/1=b/(1-sa^2)$ and can compute bisa (quadrance of the angle bisector).

```
>&factor(ratsimp(b/(1-sa2))); bisa &= %; $bisa
```

$$\frac{2b(b+a)}{\sqrt{b}\sqrt{b+a} + b+a}$$

Let us check this formula for our Egyptian values.

```
>sqrt(mxmeval("at(bisa, [a=3^2,b=4^2])")), distance(A,P)
```

4.21637021356

4.21637021356

We can also compute P using the spread formula.

```
>py&=factor(ratsimp(sa2*bisa)); $py
```

$$-\frac{b \left(\sqrt{b} \sqrt{b+a}-b-a\right)}{\sqrt{b} \sqrt{b+a}+b+a}$$

The value is the same we got with trigonometric formulas.

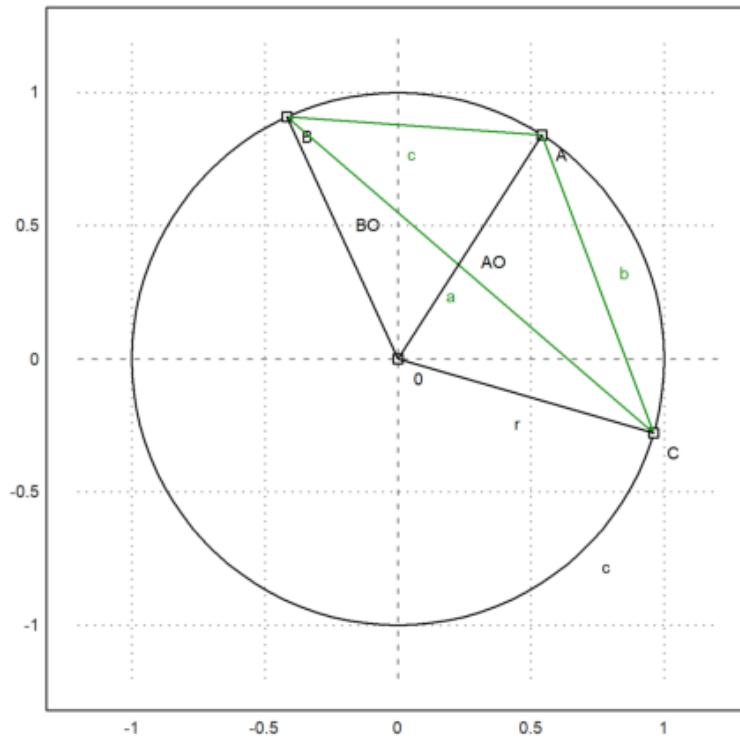
```
>sqrt(mxmeval("at(py, [a=3^2,b=4^2]))")
```

1.33333333333

The Chord Angle

Have a look at the following situation.

```
>setPlotRange(1.2); ...
>color(1); plotCircle(circleWithCenter([0,0],1)); ...
>A:=[cos(1),sin(1)]; B:=[cos(2),sin(2)]; C:=[cos(6),sin(6)]; ...
>plotPoint(A,"A"); plotPoint(B,"B"); plotPoint(C,"C"); ...
>color(3); plotSegment(A,B,"c"); plotSegment(A,C,"b"); plotSegment(C,B,"a");
>color(1); O:=[0,0]; plotPoint(O,"O"); ...
>plotSegment(A,O); plotSegment(B,O); plotSegment(C,O,"r"); ...
>insimg;
```



We can use Maxima to solve the triple spread formula for the angles at the center O for r. Thus we get a formula for the quadratic radius of the pericircle in terms of quadrances of the sides.

This time, Maxima produces some complex zeros, which we ignore.

```
>&remvalue(a,b,c,r); // hapus nilai-nilai sebelumnya untuk perhitungan baru
>rabc &= rhs(solve(triplespread(spread(b,r,r),spread(a,r,r),spread(c,r,r)),
```

$$-\frac{abc}{c^2 - 2bc + a(-2c - 2b) + b^2 + a^2}$$

We can make that an Euler function.

```
>function periradius(a,b,c) &= rabc;
```

Let us check the result for our points A,B,C.

```
>a:=quadrance(B,C); b:=quadrance(A,C); c:=quadrance(A,B);
```

The radius is indeed 1.

```
>periradius(a,b,c)
```

1

The fact is, that the spread CBA depends only on b and c. This is the chord angle theorem.

```
>$spread(b,a,c)*rabc | ratsimp
```

$$\frac{b}{4}$$

In fact the spread is $b/(4r)$, and we see that the chord angle of the chord b is half the center angle.

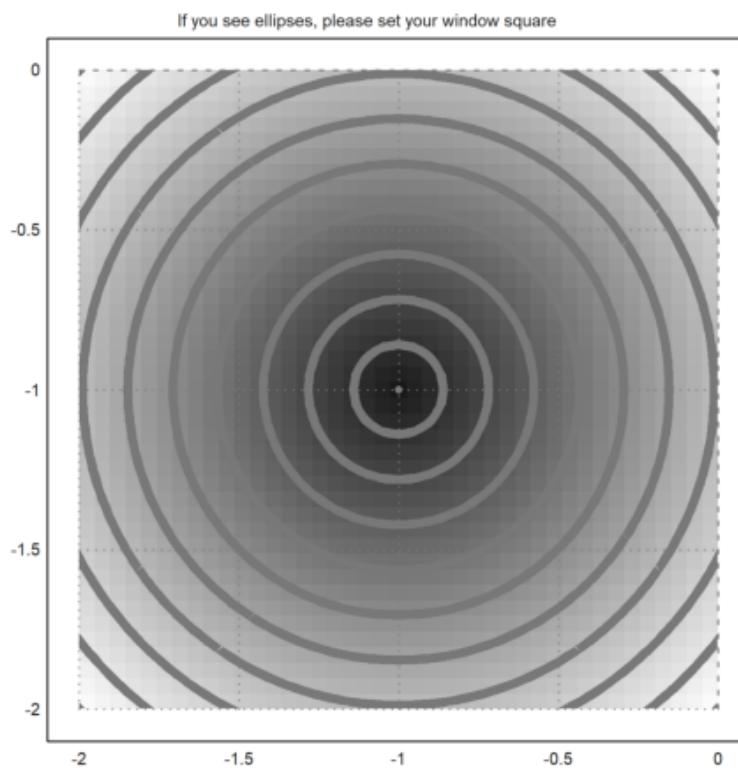
```
>$doublespread(b/(4*r))-spread(b,r,r) | ratsimp
```

0

Preliminary remark

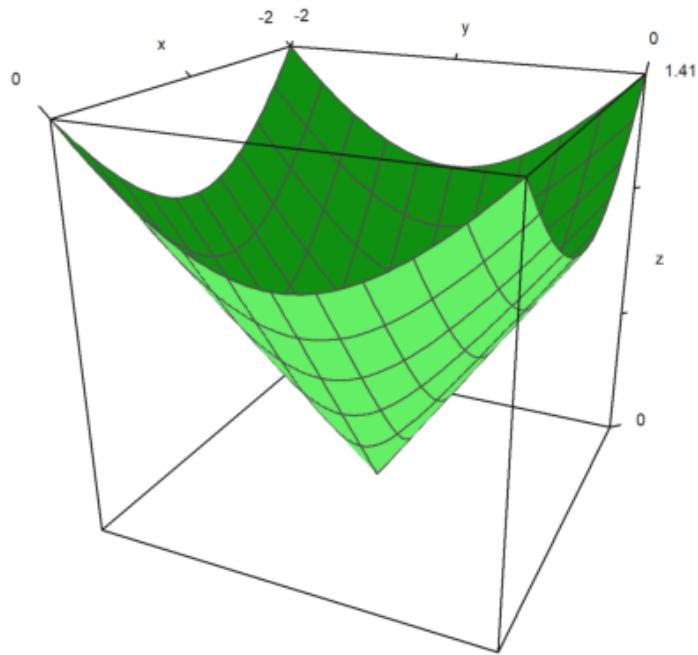
The function which, to a point M in the plane, assigns the distance AM between a fixed point A and M, has rather simple level lines: circles centered in A.

```
>&remvalue();
>A=[-1,-1];
>function d1(x,y):=sqrt((x-A[1])^2+(y-A[2])^2)
>fcontour("d1",xmin=-2,xmax=0,ymin=-2,ymax=0,hue=1, ...
>title="If you see ellipses, please set your window square"):
```



and the graph is rather simple too: the upper part of a cone:

```
>plot3d("d1",xmin=-2,xmax=0,ymin=-2,ymax=0):
```

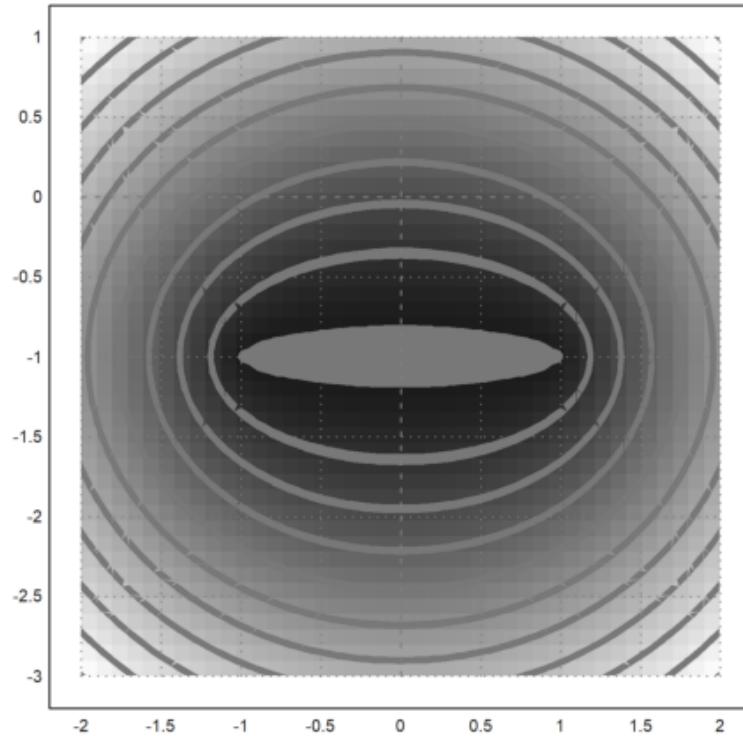


Of course the minimum 0 is attained in A.

Two points

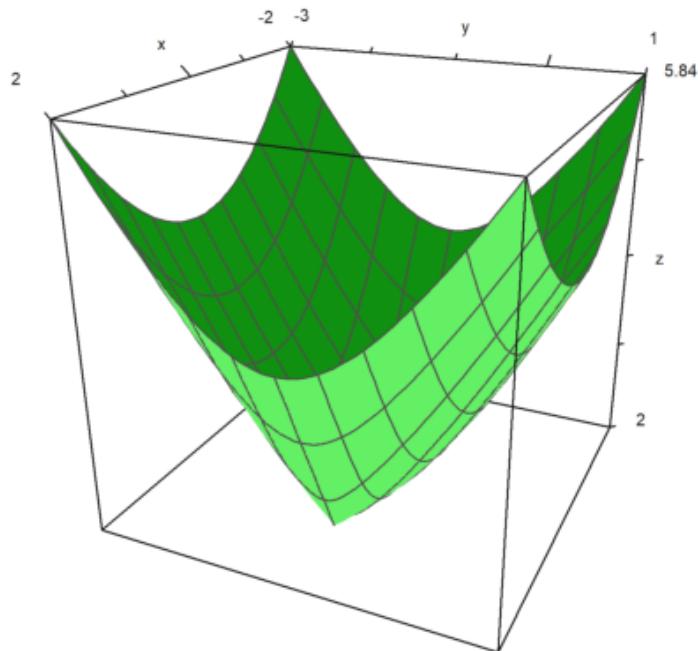
Now we look at the function $MA+MB$ where A and B are two points (fixed). It is a "well-known fact" that the level curves are ellipses, the focal points being A and B; except for the minimum AB which is constant on the segment $[AB]$:

```
>B=[1,-1];
>function d2(x,y):=d1(x,y)+sqrt((x-B[1])^2+(y-B[2])^2)
>fcontour("d2",xmin=-2,xmax=2,ymin=-3,ymax=1,hue=1):
```



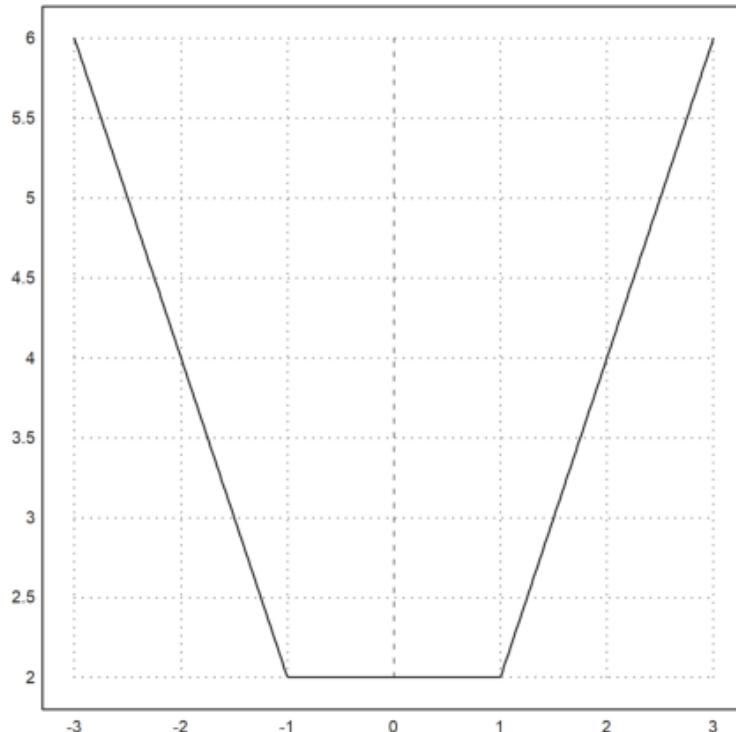
The graph is more interesting:

```
>plot3d("d2", xmin=-2, xmax=2, ymin=-3, ymax=1) :
```



The restriction to line (AB) is more famous:

```
>plot2d("abs(x+1)+abs(x-1)",xmin=-3,xmax=3):
```



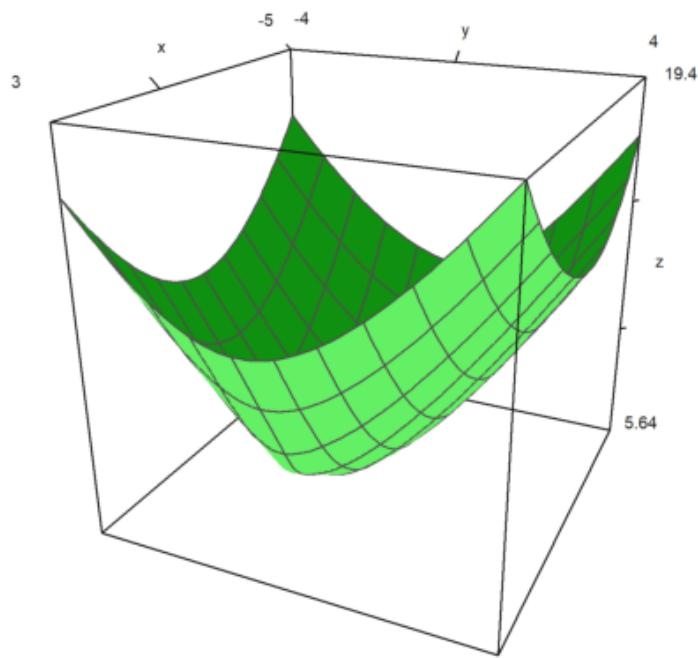
Three points

Now things are less simple: It is a little less well-known that $MA+MB+MC$ attains its minimum at one point of the plane but to determine it is less simple:

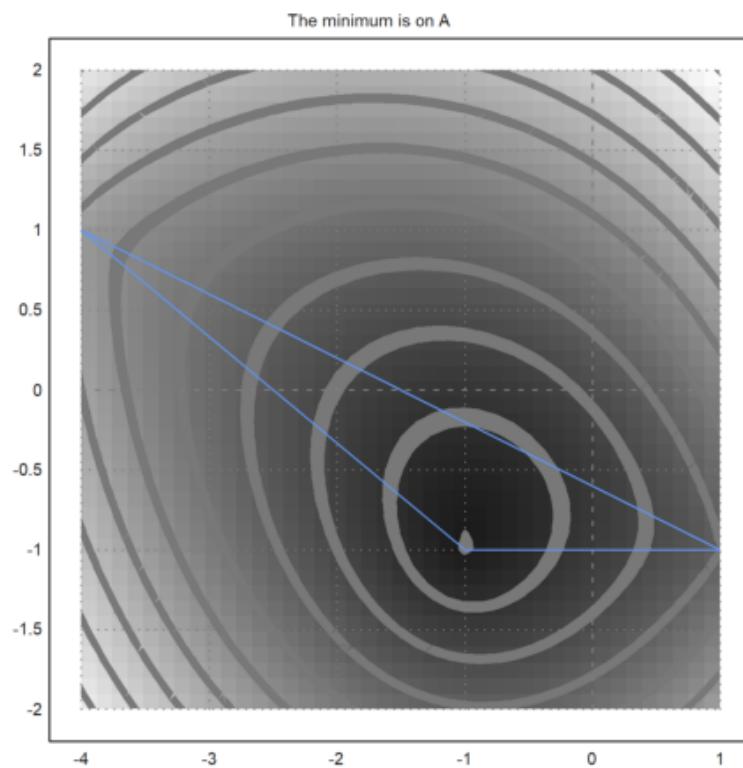
- 1) If one of the angles of the triangle ABC is more than 120° (say in A), then the minimum is attained at this very point (say $AB+AC$).

Example:

```
>C=[-4,1];
>function d3(x,y):=d2(x,y)+sqrt((x-C[1])^2+(y-C[2])^2)
>plot3d("d3",xmin=-5,xmax=3,ymin=-4,ymax=4);
>insimg;
```

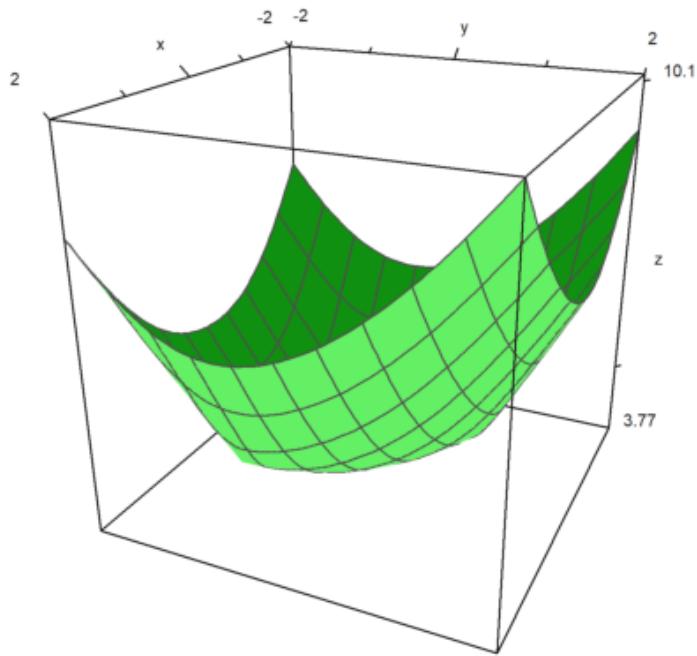


```
>fcontour("d3",xmin=-4,xmax=1,ymin=-2,ymax=2,hue=1,title="The minimum is on  
>P=(A_B_C_A)'; plot2d(P[1],P[2],add=1,color=12);  
>insimg;
```

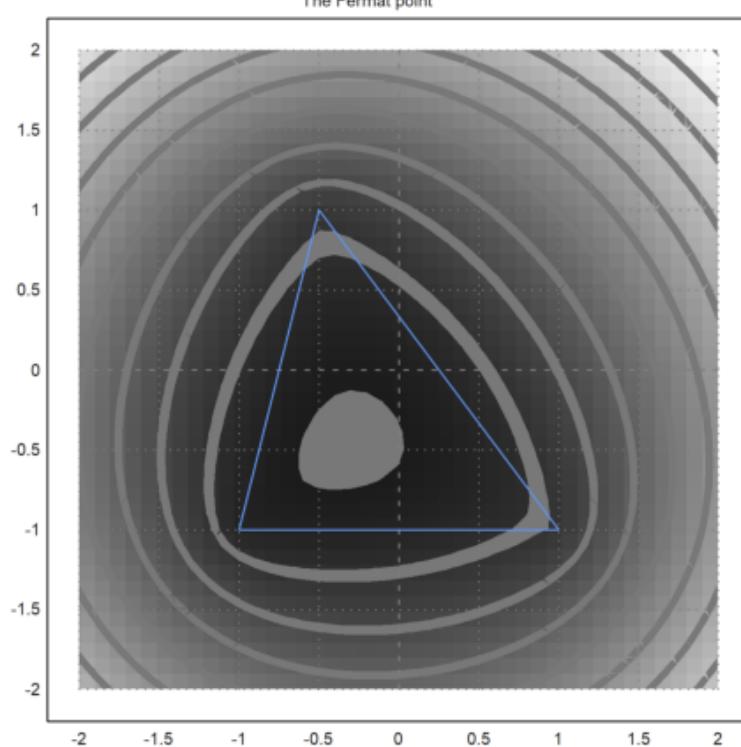


2) But if all the angles of triangle ABC are less than 120° , the minimum is on a point F in the interior of the triangle, which is the only point which sees the sides of ABC with the same angles (then 120° each):

```
>C=[-0.5,1];
>plot3d("d3",xmin=-2,xmax=2,ymin=-2,ymax=2):
```



```
>fcontour("d3",xmin=-2,xmax=2,ymin=-2,ymax=2,hue=1,title="The Fermat point"
>P=(A_B_C_A)';
>insimsg;
```



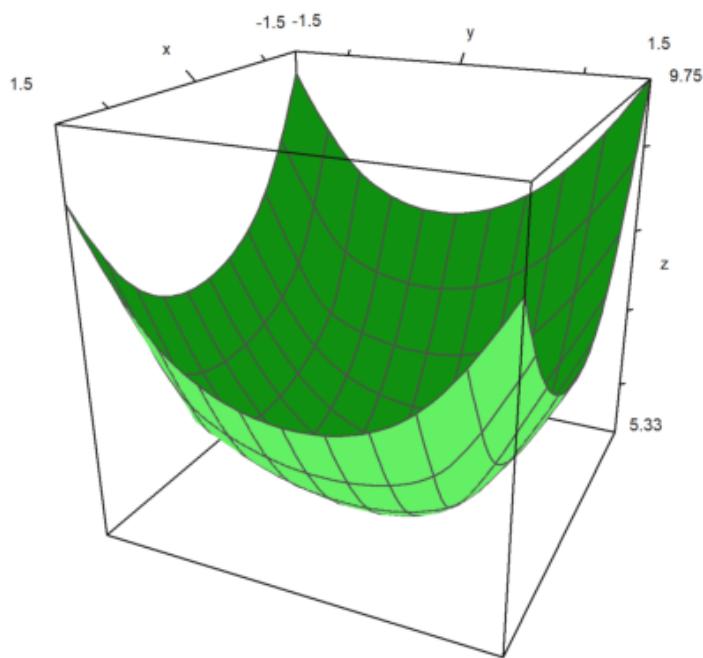
It is an interesting activity to realize the above figure with a geometry software; for example, I know a soft written in Java which has a "contour lines" instruction...

All of this above have been discovered by a french judge called Pierre de Fermat; he wrote letters to other dilettants like the priest Marin Mersenne and Blaise Pascal who worked at the income taxes. So the unique point F such that $FA+FB+FC$ is minimal, is called the Fermat point of the triangle. But it seems that a few years before, the italian Torriccelli had found this point before Fermat did! Anyway the tradition is to note this point F...

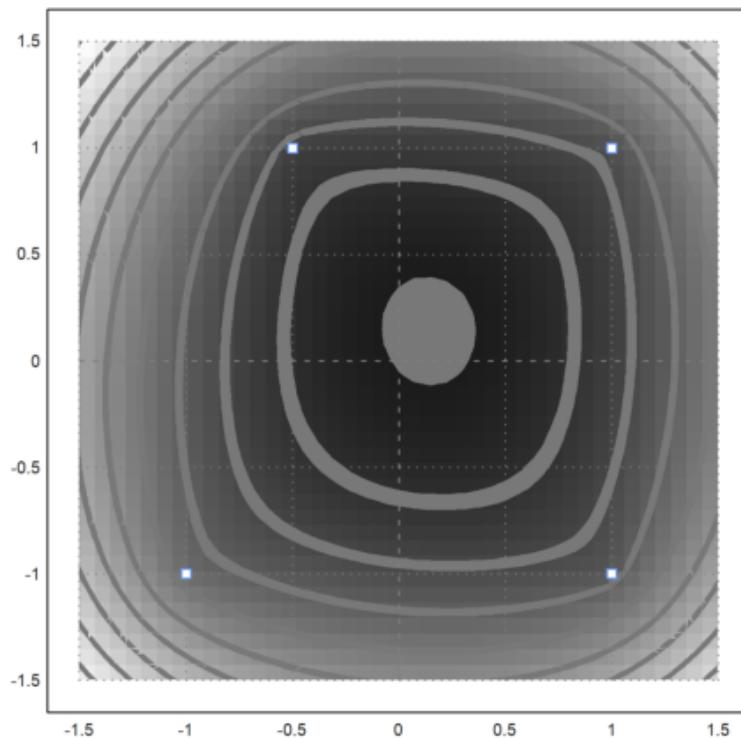
Four points

The next step is to add a 4th point D and try and minimize $MA+MB+MC+MD$; say that you are a cable TV operator and want to find in which field you must put your antenna so that you can feed four villages and use as little cable length as possible!

```
>D=[1,1];
>function d4(x,y):=d3(x,y)+sqrt((x-D[1])^2+(y-D[2])^2)
>plot3d("d4",xmin=-1.5,xmax=1.5,ymin=-1.5,ymax=1.5):
```



```
>fcontour("d4",xmin=-1.5,xmax=1.5,ymin=-1.5,ymax=1.5,hue=1);  
>P=(A_B_C_D)'; plot2d(P[1],P[2],points=1,add=1,color=12);  
>insimsg;
```



There is still a minimum and it is attained at none of the vertices A, B, C nor D:

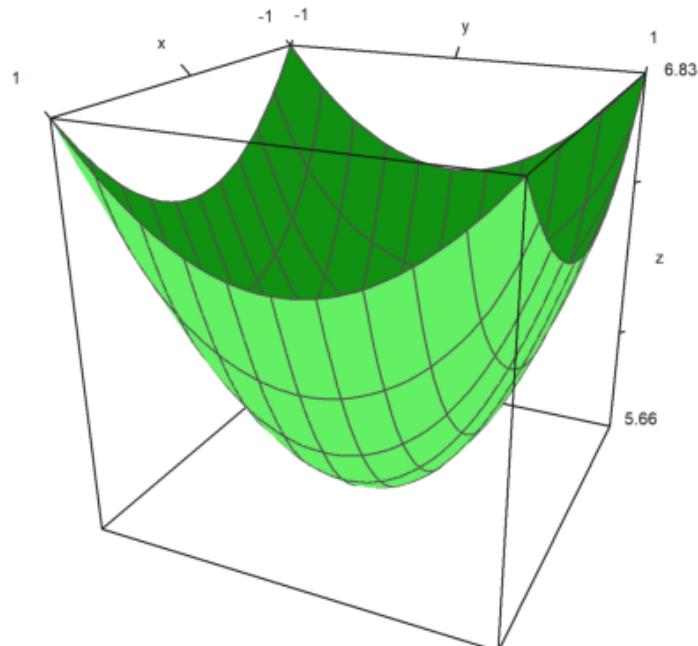
```
>function f(x):=d4(x[1],x[2])
>neldermin("f", [0.2,0.2])
```

[0.142858, 0.142857]

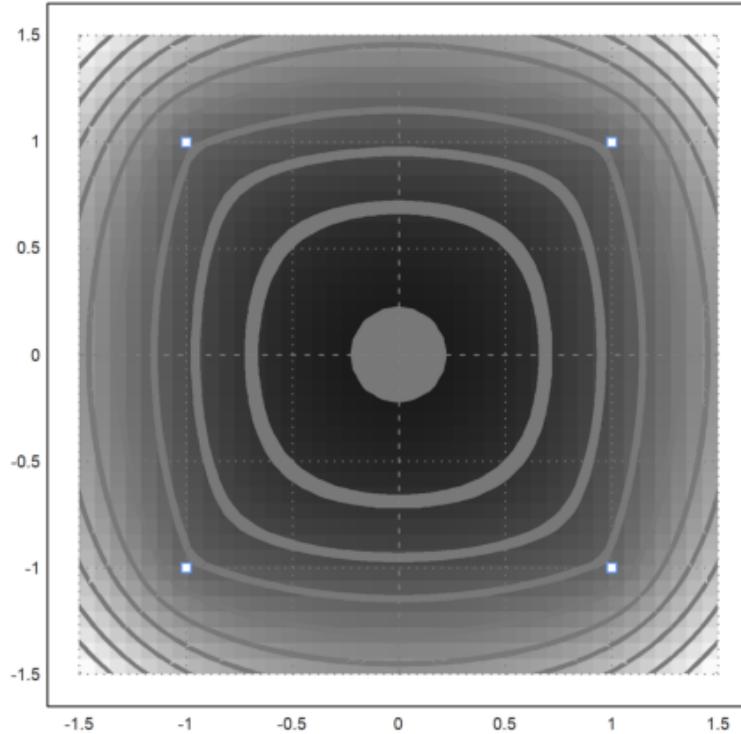
It seems that in this case, the coordinates of the optimal point are rational or near-rational...

Now ABCD is a square we expect that the optimal point will be the center of ABCD:

```
>C=[-1,1];
>plot3d("d4", xmin=-1, xmax=1, ymin=-1, ymax=1) :
```



```
>fcontour("d4", xmin=-1.5, xmax=1.5, ymin=-1.5, ymax=1.5, hue=1);
>P=(A_B_C_D)'; plot2d(P[1],P[2],add=1,color=12,points=1);
>insimg;
```



Contoh 7: Bola Dandelin dengan Povray

You can run this demonstration, if you have Povray installed, and `pvengine.exe` in the program path.

First we compute the radii of the spheres.

If you look at the figure below, you see that we need two circles touching the two lines which form the cone, and one line which forms the plane cutting the cone.

We use the `geometry.e` file of Euler for this.

```
>load geometry;
```

First the two lines forming the cone.

```
>g1 &= lineThrough([0,0],[1,a])
```

```
[- a, 1, 0]
```

```
>g2 &= lineThrough([0,0],[-1,a])
```

$[- \ a, - 1, 0]$

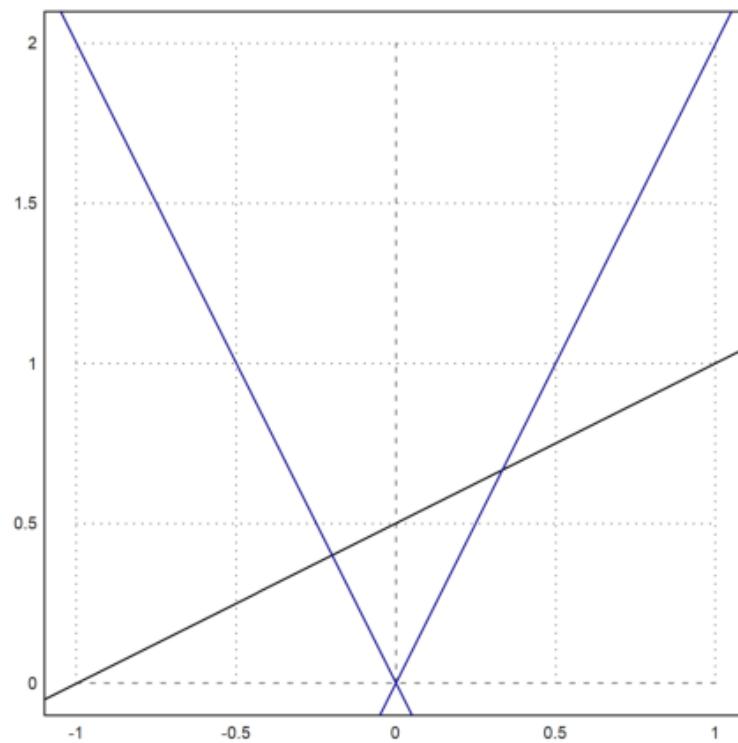
Thenm a third line.

```
>g &= lineThrough([-1,0],[1,1])
```

$[- 1, 2, 1]$

We plot everything so far.

```
>setPlotRange(-1,1,0,2);
>color(black); plotLine(g(), "")
>a:=2; color(blue); plotLine(g1(), ""), plotLine(g2(), ""):
```



Now we take a general point on the y-axis.

```
>P &= [0,u]
```

$[0, u]$

Compute the distance to g1.

```
>d1 &= distance(P,projectToLine(P,g1)); $d1
```

$$\sqrt{\left(\frac{a^2 u}{a^2 + 1} - u\right)^2 + \frac{a^2 u^2}{(a^2 + 1)^2}}$$

Compute the distance to g.

```
>d &= distance(P,projectToLine(P,g)); $d
```

$$\sqrt{\left(\frac{u + 2}{5} - u\right)^2 + \frac{(2u - 1)^2}{25}}$$

And find the centers of the two circles, where the distances are equal.

```
>sol &= solve(d1^2=d^2,u); $sol
```

$$\left[u = \frac{-\sqrt{5}\sqrt{a^2 + 1} + 2a^2 + 2}{4a^2 - 1}, u = \frac{\sqrt{5}\sqrt{a^2 + 1} + 2a^2 + 2}{4a^2 - 1} \right]$$

There are two solutions.

We evaluate the symbolic solutions, and find both centers, and both distances.

```
>u := sol()
```

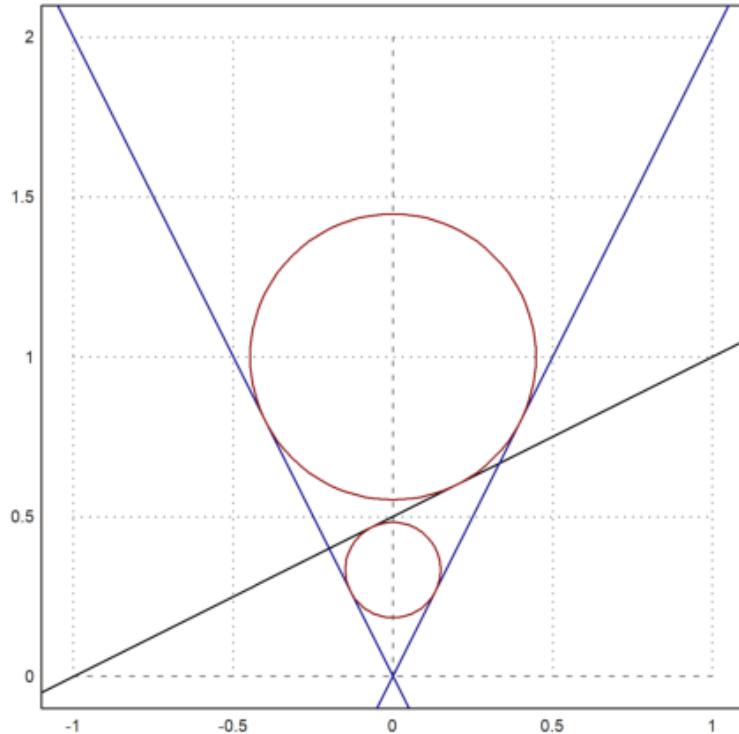
[0.333333, 1]

```
>dd := d()
```

[0.149071, 0.447214]

Plot the circles into the figure.

```
>color(red);
>plotCircle(circleWithCenter([0,u[1]],dd[1]), "");
>plotCircle(circleWithCenter([0,u[2]],dd[2]), "");
>insimg;
```



Plot with Povray

Next we plot everything with Povray. Note that you change any command in the following sequence of Povray commands, and rerun all commands with Shift-Return.

First we load the povray functions.

```
>load povray;
>defaultpovray="C:\Program Files\POV-Ray\v3.7\bin\pvengine.exe"
```

C:\Program Files\POV-Ray\v3.7\bin\pvengine.exe

We setup the scene appropriately.

```
>povstart(zoom=11,center=[0,0,0.5],height=10°,angle=140°);
```

Next we write the two spheres to the Povray file.

```
>writeln(povsphere([0,0,u[1]],dd[1],povlook(red)));
>writeln(povsphere([0,0,u[2]],dd[2],povlook(red)));
```

And the cone, transparent.

```
>writeln(povcone([0,0,0],0,[0,0,a],1,povlook(lightgray,1)));
```

We generate a plane restricted to the cone.

```
>gp=g();
>pc=povcone([0,0,0],0,[0,0,a],1,"");
>vp=[gp[1],0,gp[2]]; dp=gp[3];
>writeln(povplane(vp,dp,povlook(blue,0.5),pc));
```

Now we generate two points on the circles, where the spheres touch the cone.

```
>function turnz(v) := return [-v[2],v[1],v[3]]
>P1=projectToLine([0,u[1]],g1()); P1=turnz([P1[1],0,P1[2]]);
>writeln(povpoint(P1,povlook(yellow)));
>P2=projectToLine([0,u[2]],g1()); P2=turnz([P2[1],0,P2[2]]);
>writeln(povpoint(P2,povlook(yellow)));
```

Then we generate the two points where the spheres touch the plane. These are the foci of the ellipse.

```
>P3=projectToLine([0,u[1]],g()); P3=[P3[1],0,P3[2]];
>writeln(povpoint(P3,povlook(yellow)));
>P4=projectToLine([0,u[2]],g()); P4=[P4[1],0,P4[2]];
>writeln(povpoint(P4,povlook(yellow)));
```

Next we compute the intersection of P1P2 with the plane.

```
>t1=scalp(vp,P1)-dp; t2=scalp(vp,P2)-dp; P5=P1+t1/(t1-t2)*(P2-P1);
>writeln(povpoint(P5,povlook(yellow)));
```

We connect the points with line segments.

```
>writeln(povsegment(P1,P2,povlook(yellow)));
>writeln(povsegment(P5,P3,povlook(yellow)));
>writeln(povsegment(P5,P4,povlook(yellow)));
```

Now we generate a gray band, where the spheres touch the cone.

```
>pcw=povcone([0,0,0],0,[0,0,a],1.01);
>pc1=povcylinder([0,0,P1[3]-defaultpointsiz/2],[0,0,P1[3]+defaultpointsiz
>writeln(povintersection([pcw,pc1],povlook(gray)));
>pc2=povcylinder([0,0,P2[3]-defaultpointsiz/2],[0,0,P2[3]+defaultpointsiz
>writeln(povintersection([pcw,pc2],povlook(gray)));
```

Start the Povray program.

```
>povend();
```

To get an Anaglyph of this we need to put everything into a scene function. This function will be used twice later.

```
>function scene () ...
```

```
global a,u,dd,g,g1,defaultpointsiz;
writeln(povsphere([0,0,u[1]],dd[1],povlook(red)));
writeln(povsphere([0,0,u[2]],dd[2],povlook(red)));
writeln(povcone([0,0,0],0,[0,0,a],1,povlook(lightgray,1)));
gp=g();
pc=povcone([0,0,0],0,[0,0,a],1,"");
vp=[gp[1],0, gp[2]]; dp=gp[3];
writeln(povplane(vp,dp,povlook(blue,0.5),pc));
P1=projectToLine([0,u[1]],g1()); P1=turnz([P1[1],0,P1[2]]);
writeln(povpoint(P1,povlook(yellow)));
P2=projectToLine([0,u[2]],g1()); P2=turnz([P2[1],0,P2[2]]);
writeln(povpoint(P2,povlook(yellow)));
```

```

P3=projectToLine([0,u[1]],g()); P3=[P3[1],0,P3[2]];
writeln(povpoint(P3,povlook(yellow)));
P4=projectToLine([0,u[2]],g()); P4=[P4[1],0,P4[2]];
writeln(povpoint(P4,povlook(yellow)));
t1=scalp(vp,P1)-dp; t2=scalp(vp,P2)-dp; P5=P1+t1/(t1-t2)*(P2-P1);
writeln(povpoint(P5,povlook(yellow)));
writeln(povsegment(P1,P2,povlook(yellow)));
writeln(povsegment(P5,P3,povlook(yellow)));
writeln(povsegment(P5,P4,povlook(yellow)));
pcw=povcone([0,0,0],0,[0,0,a],1.01);
pc1=povcylinder([0,0,P1[3]-defaultpointsizew/2],[0,0,P1[3]+defaultpointsizew/2],0.01);
writeln(povintersection([pcw,pc1],povlook(gray)));
pc2=povcylinder([0,0,P2[3]-defaultpointsizew/2],[0,0,P2[3]+defaultpointsizew/2],0.01);
writeln(povintersection([pcw,pc2],povlook(gray)));
endfunction

```

You need red/cyan glasses to appreciate the following effect.

```
>povanaglyph("scene", zoom=11, center=[0,0,0.5], height=10°, angle=140°);
```

Contoh 8: Geometri Bumi

In this notebook, we want to do some spherical computations. The functions are contained in the file "spherical.e" in the examples folder. We need to load that file first.

```
>load "spherical.e";
```

To enter a geographical position, we use a vector with two coordinates in radians (north and east, negative values for south and west). The following are the coordinates for the Campus of the FMIPA UNY.

```
>FMIPA=[rad(-7,-46.467),rad(110,23.05)]
```

$[-0.13569, 1.92657]$

You can print this position with sposprint (spherical position print).

```
>sposprint(FMIPA) // posisi garis lintang dan garis bujur FMIPA UNY
```

S $7^{\circ}46.467'$ E $110^{\circ}23.050'$

Let us add two more towns, Solo and Semarang.

```
>Solo=[rad(-7,-34.333),rad(110,49.683)]; Semarang=[rad(-6,-59.05),rad(110,2
```

```
S 7°34.333' E 110°49.683'  
S 6°59.050' E 110°24.533'
```

First we compute the vector from one to the other on an ideal ball. This vector is [heading,distance] in radians. To compute the distance on the earth, we multiply with the earth radius at a latitude of 7° .

```
>br=svector(FMIPA,Solo); degprint(br[1]), br[2]*rearth(7°)->km // perkiraan
```

```
65°20'26.60''  
53.8945384608
```

This is a good approximation. The following routines use even better approximations. On such a short distance the result is almost the same.

```
>esdist(FMIPA,Semarang)->" km" // perkiraan jarak FMIPA-Semarang
```

```
Commands must be separated by semicolon or comma!  
Found: // perkiraan jarak FMIPA-Semarang (character 32)  
You can disable this in the Options menu.  
Error in:  
esdist(FMIPA,Semarang)->" km" // perkiraan jarak FMIPA-Semaran ...  
^
```

There is a function for the heading, taking the elliptical shape of the earth into account. Again, we print in an advanced way.

```
>sdegprint(esdir(FMIPA,Solo))
```

```
65.34°
```

The angle of a triangle exceeds 180° on the sphere.

```
>asum=sangle(Solo,FMIPA,Semarang)+sangle(FMIPA,Solo,Semarang)+sangle(FMIPA,
```

```
180°0'10.77''
```

This can be used to compute the area of the triangle. Note: For small triangles, this is not accurate due to the subtraction error in `asum-pi`.

```
> (asum-pi)*rearth(48°)^2->" km^2" // perkiraan luas segitiga FMIPA-Solo-Sem
```

Commands must be separated by semicolon or comma!

Found: // perkiraan luas segitiga FMIPA-Solo-Semarang (character 32)
You can disable this in the Options menu.

Error in:

```
(asum-pi)*rearth(48°)^2->" km^2" // perkiraan luas segitiga FM ...  
^
```

There is a function for this, which uses the mean latitude of the triangle to compute the earth radius, and takes care of rounding errors for very small triangles.

```
>esarea(Solo,FMIPA,Semarang)->" km^2", //perkiraan yang sama dengan fungsi
```

2123.64310526 km²

We can also add vectors to positions. A vector contains the heading and the distance, both in radians. To get a vector, we use `svector`. To add a vector to a position, we use `saddvector`.

```
>v=svector(FMIPA,Solo); sposprint(saddvector(FMIPA,v)), sposprint(Solo),
```

S 7°34.333' E 110°49.683'
S 7°34.333' E 110°49.683'

These functions assume an ideal ball. The same on the earth.

```
>sposprint(esadd(FMIPA,esdir(FMIPA,Solo),esdist(FMIPA,Solo))), sposprint(So
```

S 7°34.333' E 110°49.683'
S 7°34.333' E 110°49.683'

Let us turn to a larger example, Tugu Jogja and Monas Jakarta (menggunakan Google Earth untuk mencari koordinatnya).

```
>Tugu=[-7.7833°,110.3661°]; Monas=[-6.175°,106.811944°];  
>sposprint(Tugu), sposprint(Monas)
```

S 7°46.998' E 110°21.966'
S 6°10.500' E 106°48.717'

According to Google Earth, the distance is 429.66km. We get a good approximation.

```
>esdist(Tugu,Monas)->" km" // perkiraan jarak Tugu Jogja - Monas Jakarta
```

431.565659488 km

The heading is the same as the one computed in Google Earth.

```
>degsprint(esdir(Tugu,Monas))
```

294°17'2.85''

However, we do no longer get the exact target position, if we add the heading and distance to the orginal position. This is so, since we do not compute the inverse function exactly, but take an approximation of the earth radius along the path.

```
>sposprint(esadd(Tugu,esdir(Tugu,Monas),esdist(Tugu,Monas)))
```

S 6°10.500' E 106°48.717'

The error is not large, however.

```
>sposprint(Monas),
```

S 6°10.500' E 106°48.717'

Of course, we cannot sail with the same heading from one destination to another, if we want to take the shortest path. Imagine, you fly NE starting at any point on the earth. Then you will spiral to the north pole. Great circles do not follow a constant heading!

The following computation shows that we are way off the correct destination, if we use the same heading during our travel.

```
>dist=esdist(Tugu,Monas); hd=esdir(Tugu,Monas);
```

Now we add 10 times one-tenth of the distance, using the heading to Monas, we got in Tugu.

```
>p=Tugu; loop 1 to 10; p=esadd(p,hd,dist/10); end;
```

The result is far off.

```
>sposprint(p), skmprint(esdist(p,Monas))
```

S $6^{\circ}11.250'$ E $106^{\circ}48.372'$
1.529km

As another example, let us take two points on the earth at the same latitude.

```
>P1=[30°,10°]; P2=[30°,50°];
```

The shortest path from P1 to P2 is not the circle of latitude 30° , but a shorter path starting 10° further north at P1.

```
>sdegprint(esdir(P1,P2))
```

79.69°

But, if we follow this compass reading, we will spiral to the north pole! So we must adjust our heading along the way. For rough purposes, we adjust it at 1/10 of the total distance.

```
>p=P1; dist=esdist(P1,P2); ...
> loop 1 to 10; dir=esdir(p,P2); sdegprint(dir), p=esadd(p,dir,dist/10); e
```

79.69°
 81.67°
 83.71°
 85.78°
 87.89°
 90.00°
 92.12°
 94.22°
 96.29°
 98.33°

The distances are not right, since we will add a bit off error, if we follow the same heading for too long.

```
>skmprint(esdist(p,P2))
```

0.203km

We get a good approximation, if we adjust our heading after each 1/100 of the total distance from Tugu to Monas.

```
>p=Tugu; dist=esdist(Tugu,Monas); ...
> loop 1 to 100; p=esadd(p,esdir(p,Monas),dist/100); end;
>skmprint(esdist(p,Monas))
```

0.000km

For navigational purposes, we can get a sequence of GPS position along the great circle to Monas with the function `navigate`.

```
>load spherical; v=navigate(Tugu,Monas,10); ...
> loop 1 to rows(v); sposprint(v[#]), end;
```

```
S 7°46.998' E 110°21.966'
S 7°37.422' E 110°0.573'
S 7°27.829' E 109°39.196'
S 7°18.219' E 109°17.834'
S 7°8.592' E 108°56.488'
S 6°58.948' E 108°35.157'
S 6°49.289' E 108°13.841'
S 6°39.614' E 107°52.539'
S 6°29.924' E 107°31.251'
S 6°20.219' E 107°9.977'
S 6°10.500' E 106°48.717'
```

We write a function, which plots the earth, the two positions, and the positions in between.

```
>function testplot ...
useglobal;
plotearth;
plotpos(Tugu,"Tugu Jogja"); plotpos(Monas,"Tugu Monas");
plotposline(v);
endfunction
```

Now plot everything.

```
>plot3d("testplot",angle=25, height=6,>own,>user,zoom=4):
```

```
>plot3d("testplot", angle=25, height=6, distance=5, own=1, anaglyph=1, zoom=4) :
```

Latihan

1. Gambarlah segi-n beraturan jika diketahui titik pusat O, n, dan jarak titik pusat ke titik-titik sudut segi-n tersebut (jari-jari lingkaran luar segi-n), r.

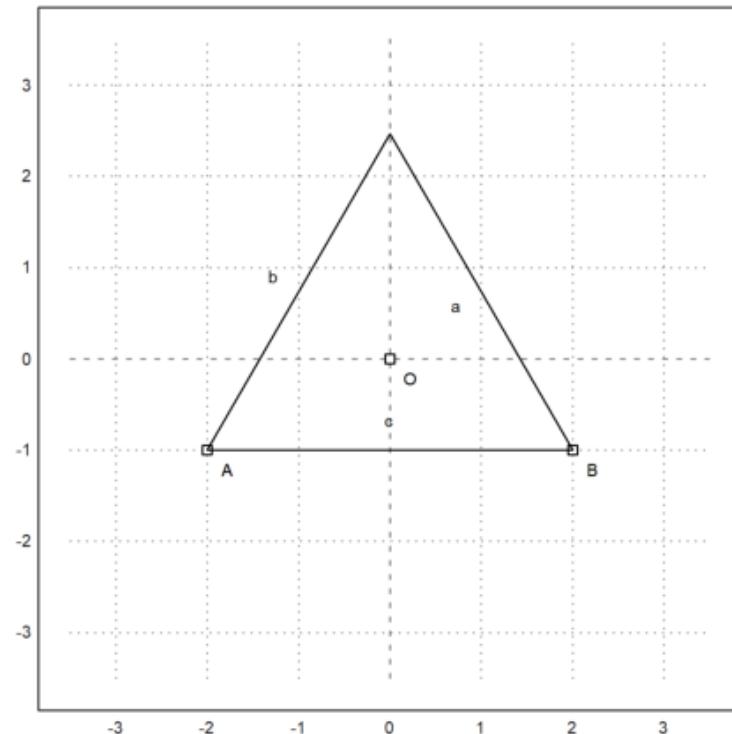
Petunjuk:

- Besar sudut pusat yang menghadap masing-masing sisi segi-n adalah $(360/n)$.
- Titik-titik sudut segi-n merupakan perpotongan lingkaran luar segi-n dan garis-garis yang melalui pusat dan saling membentuk sudut sebesar kelipatan $(360/n)$.
- Untuk n ganjil, pilih salah satu titik sudut adalah di atas.
- Untuk n genap, pilih 2 titik di kanan dan kiri lurus dengan titik pusat.
- Anda dapat menggambar segi-3, 4, 5, 6, 7, dst beraturan.

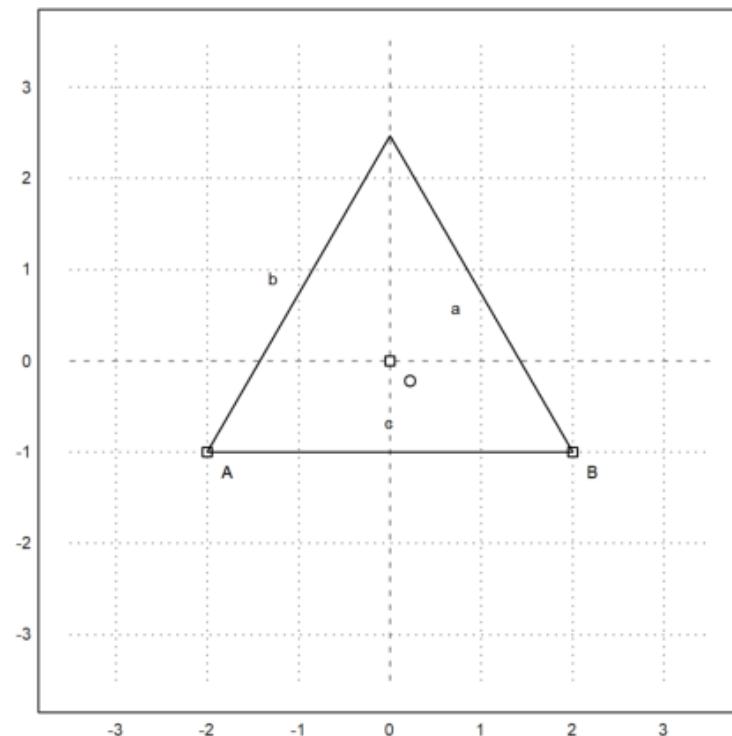
```
>load geometry
```

Numerical and symbolic geometry.

```
>setPlotRange (-3.5,3.5,-3.5,3.5);
>O=[0,0]; plotPoint(O,"O");
>A=[-2,-1]; plotPoint(A,"A");
>B=[2,-1]; plotPoint(B,"B");
>C=[0,2*3^0.5-1]; plotPoint(C,"C");
>plotSegment(A,B,"c");
>plotSegment(B,C,"a");
>plotSegment(A,C,"b");
>aspect(1);
```



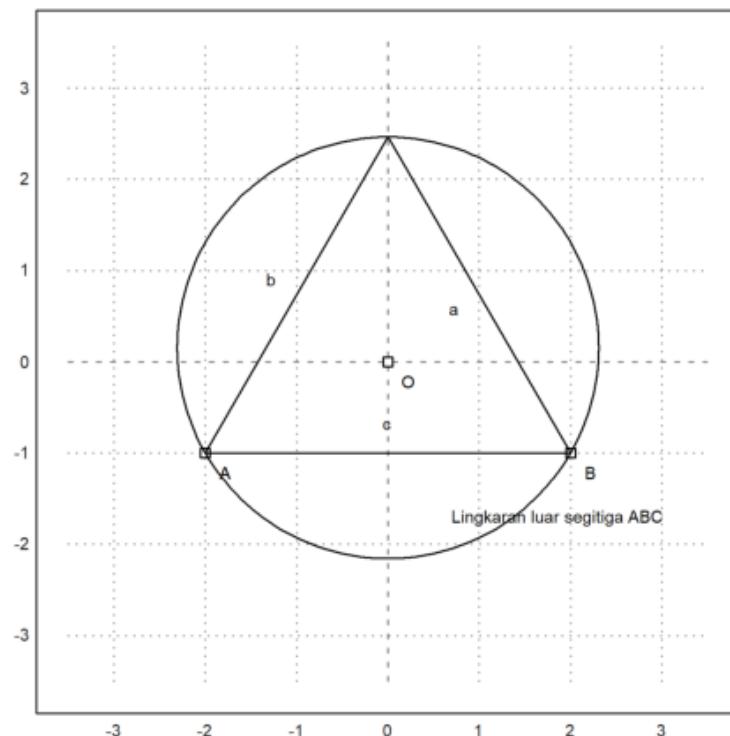
```
>c=circleThrough(A,B,C) :
```



```
>R=getCircleRadius(c);
>O=getCircleCenter(c)
```

[0, 0.154701]

```
>plotCircle(c,"Lingkaran luar segitiga ABC");
```

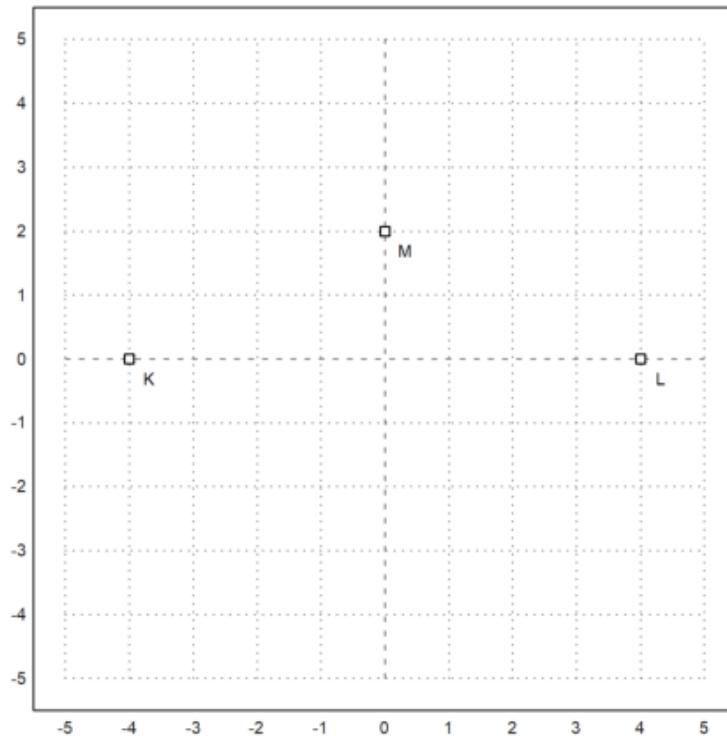


2. Gambarlah suatu parabola yang melalui 3 titik yang diketahui.

Petunjuk:

- Misalkan persamaan parabolanya $y = ax^2 + bx + c$.
- Substitusikan koordinat titik-titik yang diketahui ke persamaan tersebut.
- Selesaikan SPL yang terbentuk untuk mendapatkan nilai-nilai a, b, c.

```
>setPlotRange(5);
>K=[-4,0]; L=[4,0] ; M=[0,2];
>plotPoint(K,"K"); plotPoint(L,"L"); plotPoint(M,"M") :
```



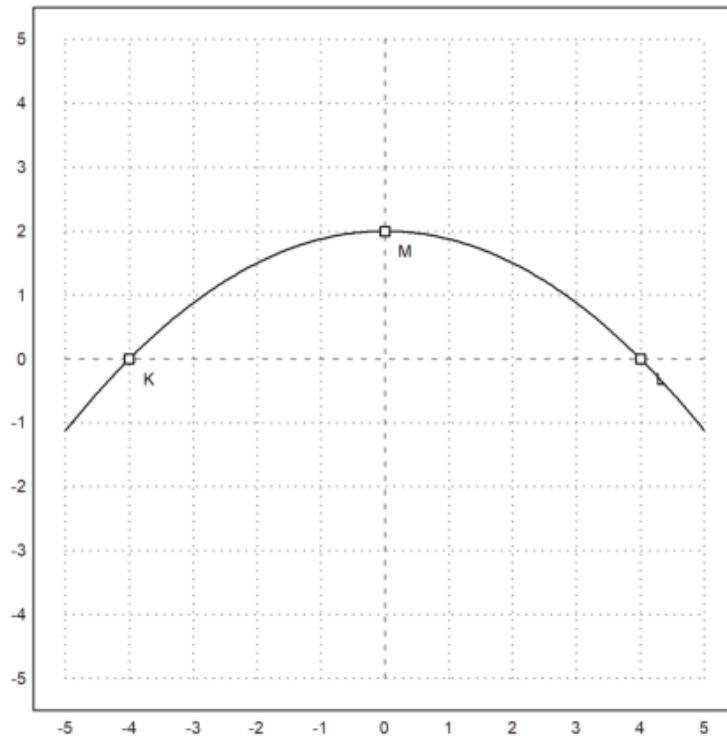
```
>sol &= solve([16*a+8*b=-c, 16*a-8*b=-c, c=2], [a, b, c])
```

$$\left[\left[a = -\frac{1}{8}, b = 0, c = 2 \right] \right]$$

```
>function y&=-1/8*(x^2)-0*x+2
```

$$\frac{x^2}{8} - \frac{1}{8}$$

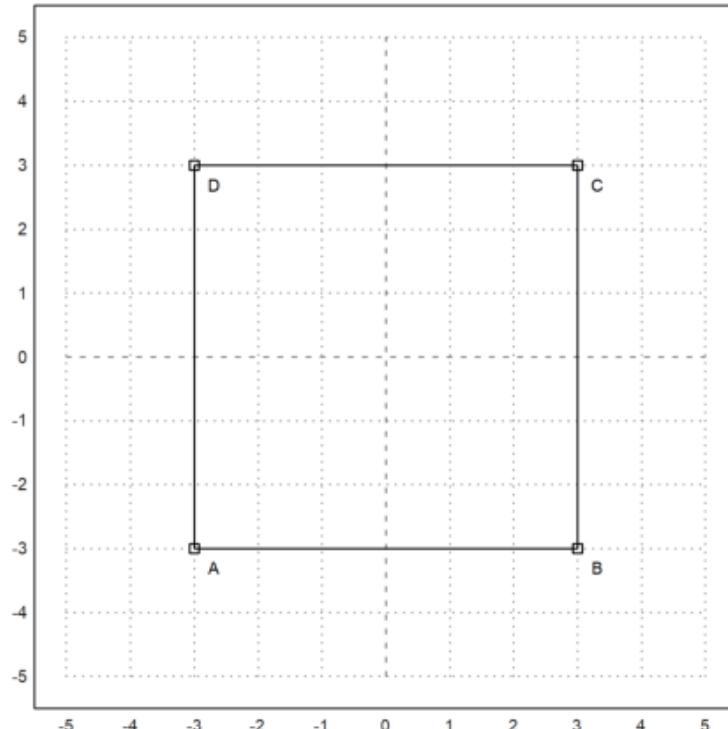
```
>plot2d("-1/8*(x^2)-0*x+2", -5, 5, -5, 5); ...
>plotPoint(K, "K"); plotPoint(L, "L"); plotPoint(M, "M");
```



3. Gambarlah suatu segi-4 yang diketahui keempat titik sudutnya, misalnya A, B, C, D.

- Tentukan apakah segi-4 tersebut merupakan segi-4 garis singgung (sisinya-sisinya merupakan garis singgung lingkaran yang sama yakni lingkaran dalam segi-4 tersebut).
- Suatu segi-4 merupakan segi-4 garis singgung apabila keempat garis bagi sudutnya bertemu di satu titik.
- Jika segi-4 tersebut merupakan segi-4 garis singgung, gambar lingkaran dalamnya.
- Tunjukkan bahwa syarat suatu segi-4 merupakan segi-4 garis singgung apabila hasil kali panjang sisi-sisi yang berhadapan sama.

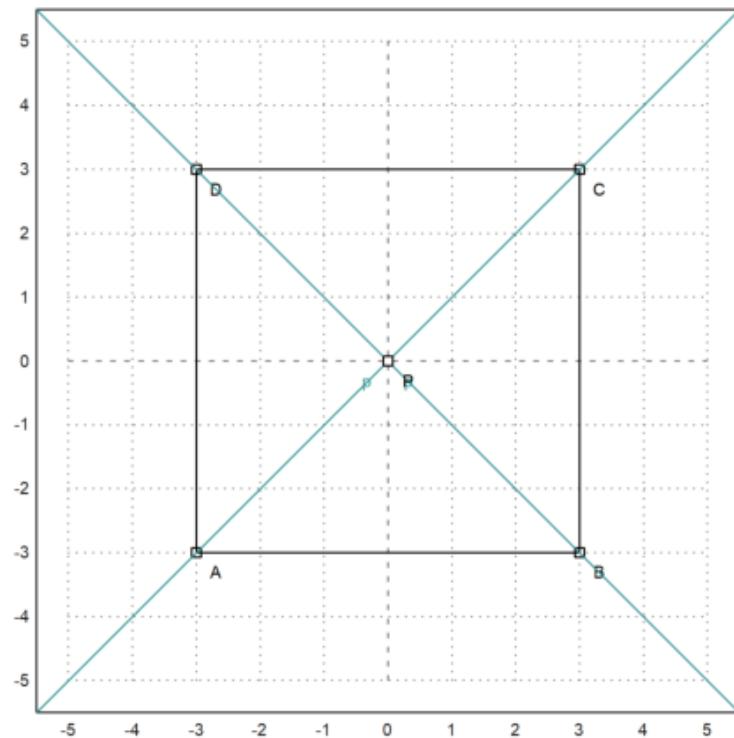
```
>setPlotRange(5);
>A=[-3,-3]; B=[3,-3] ; C=[3,3]; D=[-3,3];
>plotPoint(A, "A"); plotPoint(B, "B"); plotPoint(C, "C"); plotPoint(D, "D");
>plotSegment(A,B,""); plotSegment(B,C,""); plotSegment(C,D,""); plotSegment(D,A,"")
```



```
>l=angleBisector(A,B,C);
>g=angleBisector(B,C,D);
>P=lineIntersection(l,g)
```

[0, 0]

```
>color(5); plotLine(l); plotLine(g); color(1);
>plotPoint(P, "P"):
```



```
>r=norm(P-projectToLine(P, lineThrough(A, B)) )
```

3

```
>plotCircle(circleWithCenter(P, r), "Lingkaran dalam segiempat ABC")
```

Dari gambar di atas, terlihat bahwa sisi-sisinya merupakan garis singgung lingkaran yang sama yaitu lingkaran dalam segiempat.

Kemudian akan ditunjukkan bahwa hasil kali panjang sisi-sisi yang berhadapan sama.

```
>AB=norm(A-B) // panjang sisi AB
```

6

```
>BC=norm(B-C) // panjang sisi ABC
```

6

```
>CD=norm(C-D) // panjang sisi CD
```

6

```
>DA=norm(D-A) // panjang sisi DA
```

6

```
>AB.CD
```

36

```
>DA.BC
```

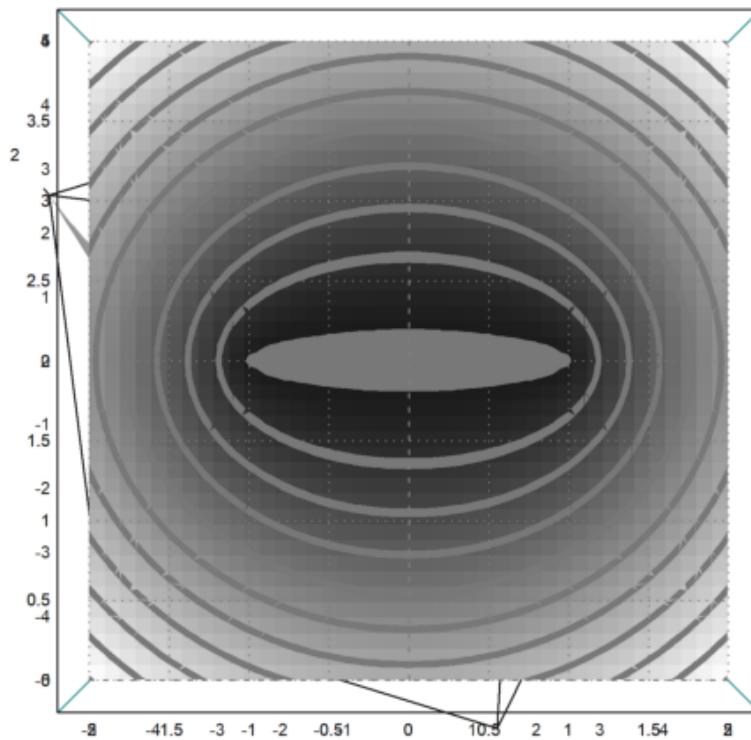
36

4. Gambarlah suatu ellips jika diketahui kedua titik fokusnya, misalnya P dan Q. Ingat ellips dengan fokus P dan Q adalah tempat kedudukan titik-titik yang jumlah jarak P dan ke Q selalu sama (konstan).

Penyelesaian:

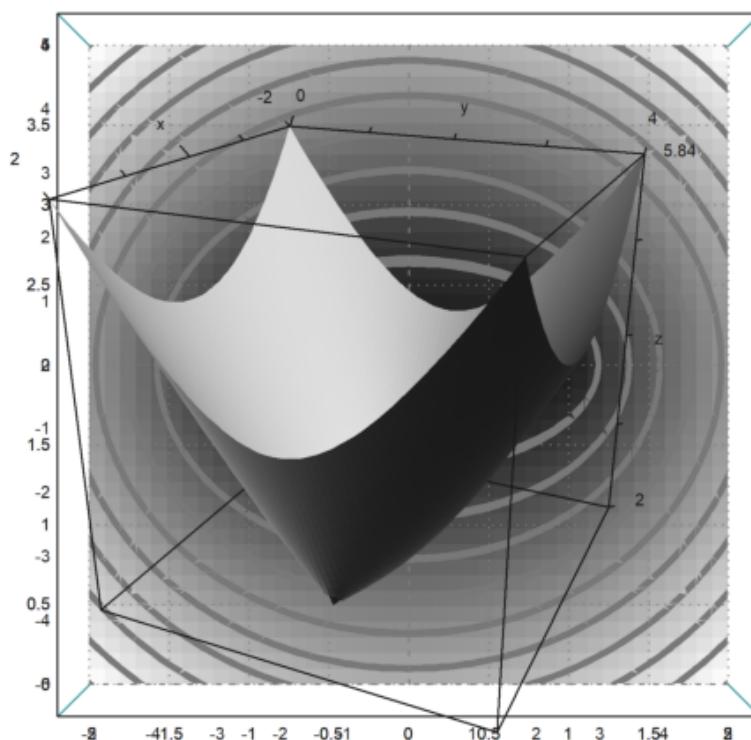
Diketahui kedua titik fokus $P=[-1,-1]$ dan $Q=[1,-1]$

```
>P=[-1,2]; Q=[1,2];
>function d1(x,y):=sqrt((x-P[1])^2+(y-P[2])^2)
>function d2(x,y):=d1(x,y)+sqrt((x-Q[1])^2+(y-Q[2])^2)
>fcontour("d2",xmin=-2,xmax=2,ymin=0,ymax=4,hue=1):
```

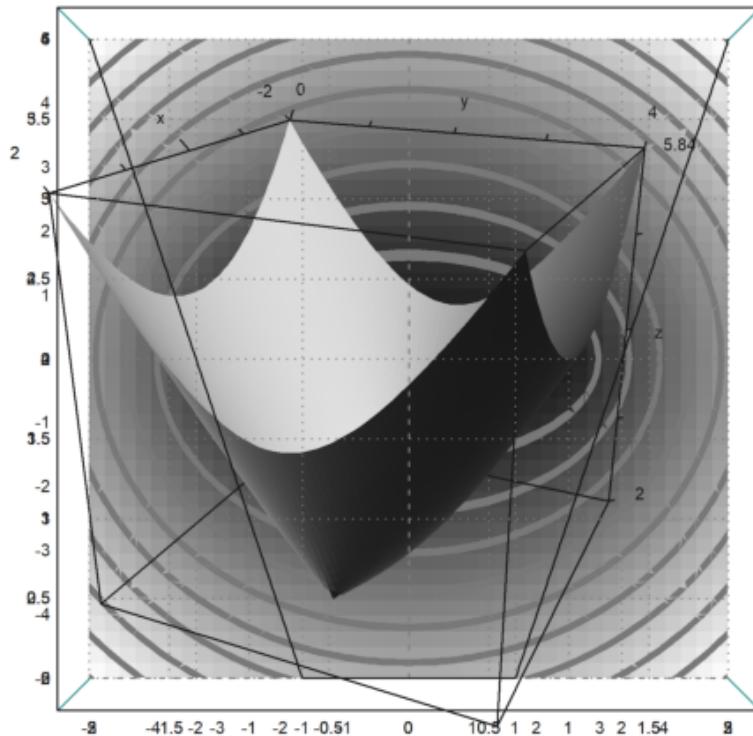


Grafik yang lebih menarik

```
>plot3d("d2", xmin=-2, xmax=2, ymin=0, ymax=4, hue=1) :
```

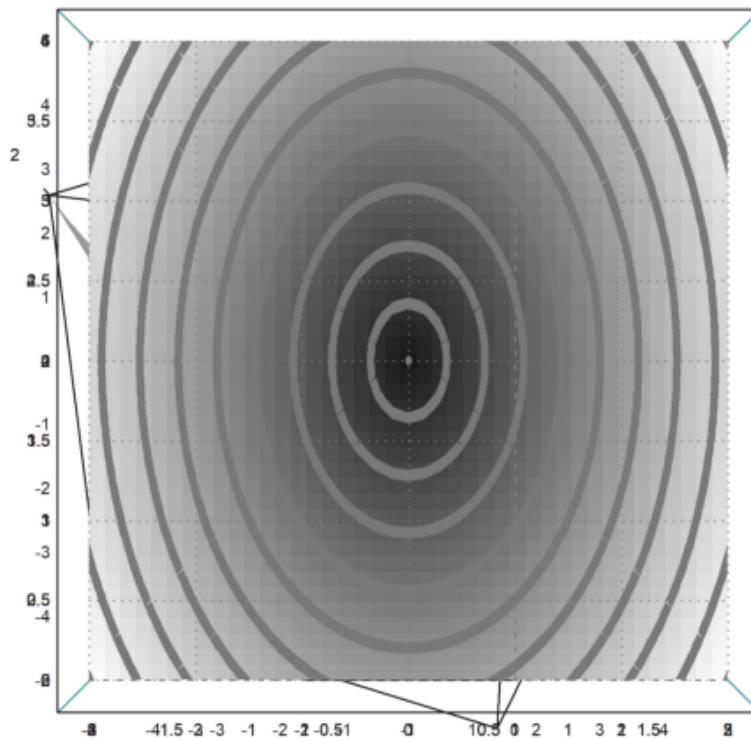


```
>plot2d("abs(x+1)+abs(x-1)",xmin=-3,xmax=3):
```



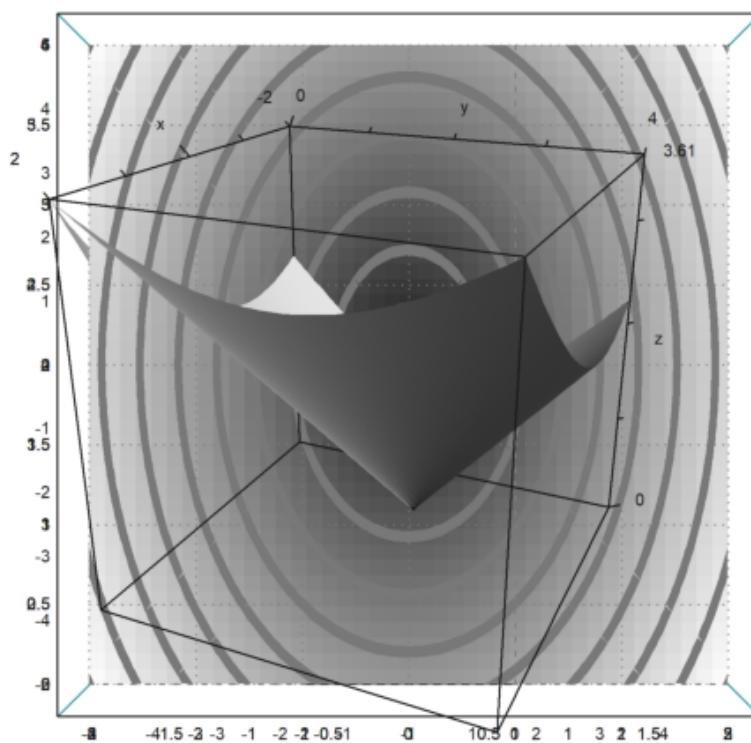
5. Gambarlah suatu hiperbola jika diketahui kedua titik fokusnya, misalnya P dan Q. Ingat ellips dengan fokus P dan Q adalah tempat kedudukan titik-titik yang selisih jarak ke P dan ke Q selalu sama (konstan).

```
>P=[-1,2]; Q=[1,2];
>function d3(x,y):=sqrt((x-P[1])^2+(y-P[2])^2)
>function d4(x,y):=d3(x,y)+sqrt((x-Q[1])^2+(y-Q[2])^2)
>fcontour("d3",xmin=-4,xmax=2,ymin=0,ymax=4,hue=1):
```

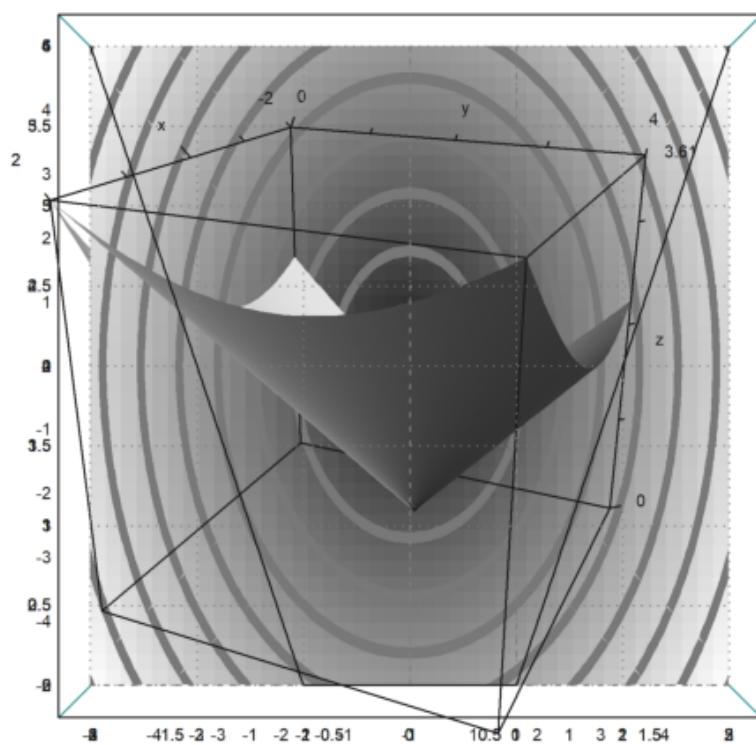


Grafik yang lebih menarik

```
>plot3d("d3", xmin=-2, xmax=2, ymin=0, ymax=4, hue=1) :
```



```
>plot2d("abs(x+1)+abs(x-1)",xmin=-3,xmax=3):
```



BAB

EMT untuk Statistika

Dalam buku catatan ini, kami mendemonstrasikan plot statistik utama, pengujian, dan distribusi di Euler.

Mari kita mulai dengan beberapa statistik deskriptif. Ini bukan pengantar statistik. Jadi, Anda mungkin memerlukan beberapa latar belakang untuk memahami detailnya.

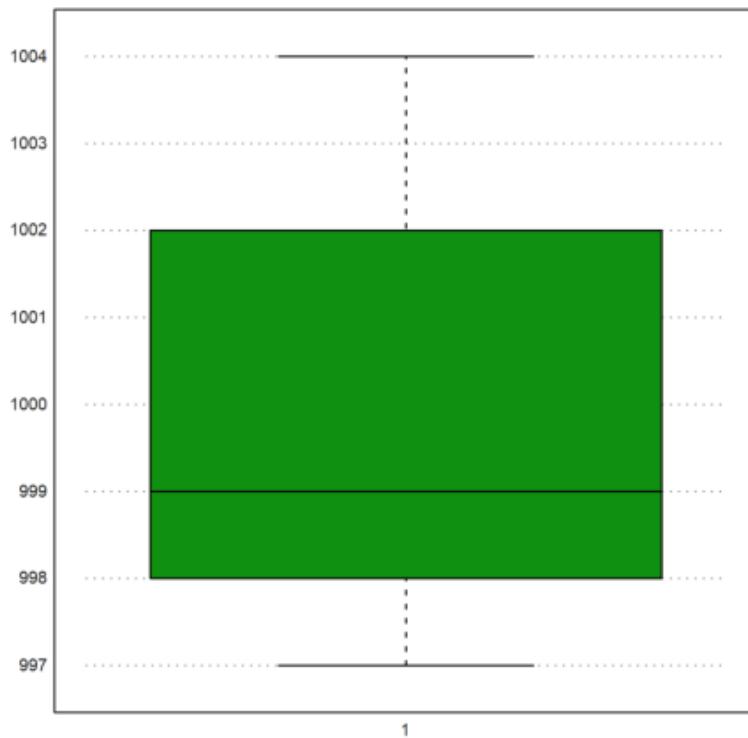
Asumsikan pengukuran berikut. Kami ingin menghitung nilai rata-rata dan standar deviasi yang diukur.

```
>M=[1000,1004,998,997,1002,1001,998,1004,998,997]; ...
>mean(M), dev(M),
```

```
999.9
2.72641400622
```

Kita dapat memplot plot kotak-dan-kumis untuk data. Dalam kasus kami tidak ada outlier.

```
>boxplot(M) :
```



Kami menghitung probabilitas bahwa suatu nilai lebih besar dari 1005, dengan asumsi nilai terukur dan distribusi normal.

Semua fungsi untuk distribusi di Euler diakhiri dengan ...dis dan menghitung distribusi probabilitas kumulatif (CPF).

$$\text{normaldis}(x, m, d) = \int_{-\infty}^x \frac{1}{d\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{t-m}{d})^2} dt.$$

Kami mencetak hasilnya dalam % dengan akurasi 2 digit menggunakan fungsi cetak.

```
>print((1-normaldis(1005, mean(M), dev(M)))*100, 2, unit=" %")
```

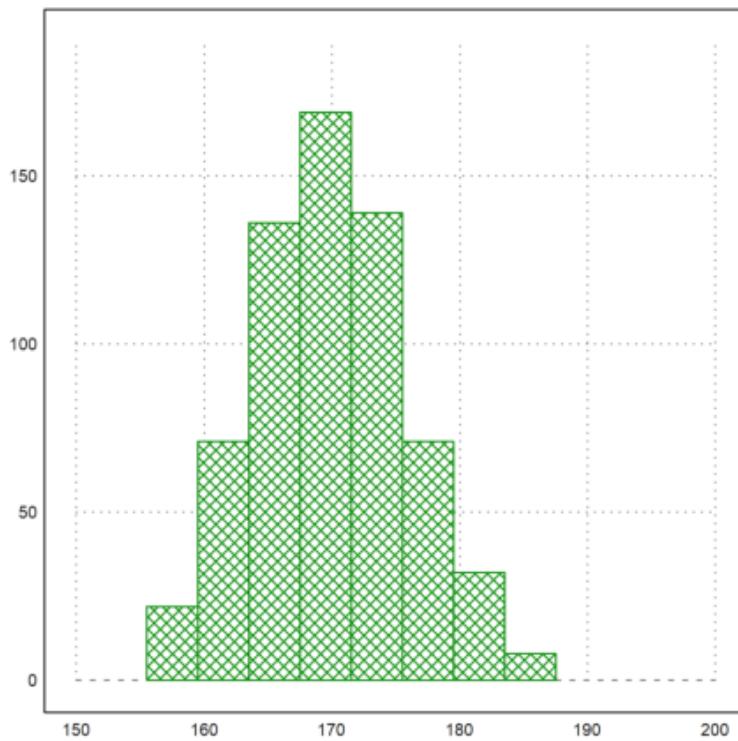
3.07 %

Untuk contoh berikutnya, kami mengasumsikan jumlah pria berikut dalam rentang ukuran yang diberikan.

```
>r=155.5:4:187.5; v=[22,71,136,169,139,71,32,8];
```

Berikut adalah plot distribusinya.

```
>plot2d(r,v,a=150,b=200,c=0,d=190,bar=1,style="/") :
```



Kita bisa memasukkan data mentah tersebut ke dalam sebuah tabel.

Tabel adalah metode untuk menyimpan data statistik. Tabel kita harus berisi tiga kolom: Awal jangkauan, akhir jangkauan, jumlah orang dalam jangkauan.

Tabel dapat dicetak dengan header. Kami menggunakan vektor string untuk mengatur header.

```
>T:=r[1:8]' | r[2:9]' | v'; writetable(T,labc=["from","to","count"])
```

from	to	count
155.5	159.5	22
159.5	163.5	71
163.5	167.5	136
167.5	171.5	169
171.5	175.5	139
175.5	179.5	71
179.5	183.5	32
183.5	187.5	8

Jika kita membutuhkan nilai rata-rata dan statistik lain dari ukuran, kita perlu menghitung titik tengah rentang. Kita dapat menggunakan dua kolom pertama dari tabel kita untuk ini. Sumbul "|" digunakan untuk memisahkan kolom, fungsi "writetable" digunakan untuk menulis tabel, dengan opsi "labc" adalah untuk menentukan header kolom.

```
>(T[,1]+T[,2])/2 // the midpoint of each interval
```

```
157.5
161.5
165.5
169.5
173.5
177.5
181.5
185.5
```

Tetapi lebih mudah, untuk melipat rentang dengan vektor [1/2.1/2].

```
>M=fold(r, [0.5, 0.5])
```

```
[157.5, 161.5, 165.5, 169.5, 173.5, 177.5, 181.5, 185.5]
```

Sekarang kita dapat menghitung rata-rata dan deviasi sampel dengan frekuensi yang diberikan.

```
>{m, d}=meandev(M, v); m, d,
```

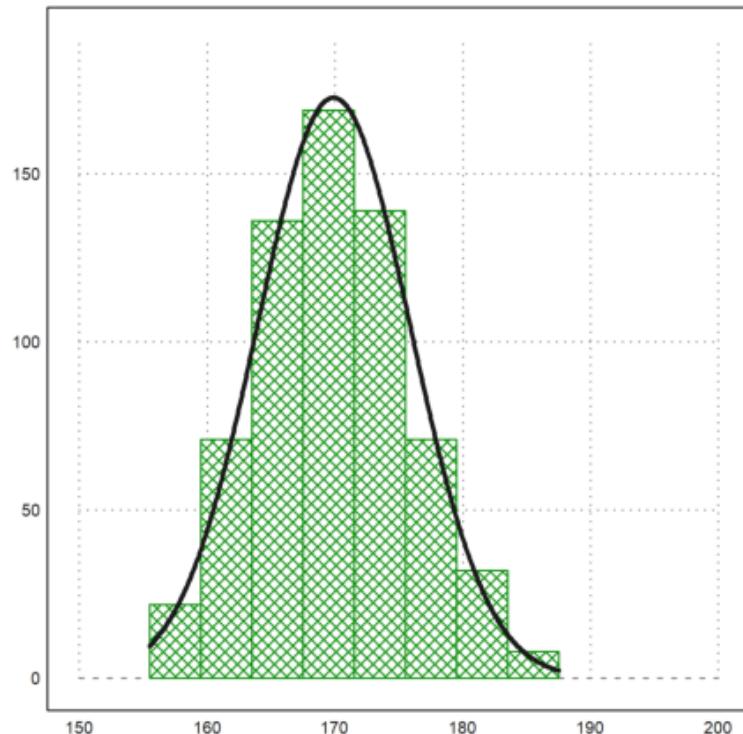
```
169.901234568
5.98912964449
```

Mari kita tambahkan distribusi normal dari nilai-nilai ke plot batang di atas. Rumus untuk distribusi normal dengan mean m dan standar deviasi d adalah:

$$y = \frac{1}{d\sqrt{2\pi}} e^{\frac{-(x-m)^2}{2d^2}}.$$

Karena nilainya antara 0 dan 1, untuk memplotnya pada bar plot harus dikalikan dengan 4 kali jumlah total data.

```
>plot2d("qnormal(x,m,d)*sum(v)*4", ...
> xmin=min(r), xmax=max(r), thickness=3, add=1):
```



Tabel

Di direktori notebook ini Anda menemukan file dengan tabel. Data tersebut mewakili hasil survei. Berikut adalah empat baris pertama dari file tersebut. Data berasal dari buku online Jerman "Einführung in die Statistik mit R" oleh A. Handl.

```
>printfile("table.dat", 4);
```

```
Could not open the file
table.dat
for reading!
Try "trace errors" to inspect local variables after errors.
printfile:
  open(filename, "r");
```

Tabel berisi 7 kolom angka atau token (string). Kami ingin membaca tabel dari file. Pertama, kami menggunakan terjemahan kami sendiri untuk token.

Untuk ini, kami mendefinisikan set token. Fungsi `strtokens()` mendapatkan vektor string token dari string yang diberikan.

```
>mf := ["m", "f"]; yn := ["y", "n"]; ev := strtokens("g vg m b vb");
```

Sekarang kita membaca tabel dengan terjemahan ini.

Argumen tok2, tok4 dll. adalah terjemahan dari kolom tabel. Argumen ini tidak ada dalam daftar parameter readtable(), jadi Anda harus menyediakannya dengan ":=".

```
>{MT,hd}=readtable("table.dat",tok2:=mf,tok4:=yn,tok5:=ev,tok7:=yn);
```

```
Could not open the file
table.dat
for reading!
Try "trace errors" to inspect local variables after errors.
readtable:
    if filename!=none then open(filename,"r"); endif;
```

```
>load over statistics;
```

Untuk mencetak, kita perlu menentukan set token yang sama. Kami mencetak empat baris pertama saja.

```
>writetable(MT[1:4],labc=hd,wc=5,tok2:=mf,tok4:=yn,tok5:=ev,tok7:=yn);
```

```
MT is not a variable!
Error in:
writetable(MT[1:4],labc=hd,wc=5,tok2:=mf,tok4:=yn,tok5:=ev,tok ...
^
```

Titik "." mewakili nilai-nilai, yang tidak tersedia.

Jika kita tidak ingin menentukan token untuk terjemahan terlebih dahulu, kita hanya perlu menentukan, kolom mana yang berisi token dan bukan angka.

```
>ctok=[2,4,5,7]; {MT,hd,tok}=readtable("table.dat",ctok=ctok);
```

```
Could not open the file
table.dat
for reading!
Try "trace errors" to inspect local variables after errors.
readtable:
    if filename!=none then open(filename,"r"); endif;
```

Fungsi readtable() sekarang mengembalikan satu set token.

```
>tok
```

Variable tok not found!
Error in:
tok ...
^

Tabel berisi entri dari file dengan token yang diterjemahkan ke angka.

String khusus NA="." ditafsirkan sebagai "Tidak Tersedia", dan mendapatkan NAN (bukan angka) dalam tabel. Terjemahan ini dapat diubah dengan parameter NA, dan NAvl.

```
>MT [1]
```

MT is not a variable!
Error in:
MT[1] ...
^

Berikut isi tabel dengan angka yang belum diterjemahkan.

```
>writetable(MT,wc=5)
```

Variable or function MT not found.
Error in:
writetable(MT,wc=5) ...
^

Untuk kenyamanan, Anda dapat memasukkan output readtable() ke dalam daftar.

```
>Table={readtable("table.dat",ctok=ctok)};
```

Could not open the file
table.dat
for reading!
Try "trace errors" to inspect local variables after errors.
readtable:
if filename!=none then open(filename,"r"); endif;

Menggunakan kolom token yang sama dan token yang dibaca dari file, kita dapat mencetak tabel. Kita dapat menentukan ctok, tok, dll. Atau menggunakan daftar Tabel.

```
>writetable(Table,ctok=ctok,wc=5);
```

Variable or function Table not found.
 Error in:
 writetable(Table,ctok=ctok,wc=5); ...
 ^

Fungsi `tablecol()` mengembalikan nilai kolom tabel, melewatkannya setiap baris dengan nilai `NAN("." dalam file)`, dan indeks kolom, yang berisi nilai-nilai ini.

```
>{c,i}=tablecol(MT,[5,6]);
```

Variable or function MT not found.
 Error in:
 {c,i}=tablecol(MT,[5,6]); ...
 ^

Kita dapat menggunakan ini untuk mengekstrak kolom dari tabel untuk tabel baru.

```
>j=[1,5,6]; writetable(MT[i,j],labc=hd[j],ctok=[2],tok=tok)
```

Variable or function i not found.
 Error in:
 j=[1,5,6]; writetable(MT[i,j],labc=hd[j],ctok=[2],tok=tok) ...
 ^

Tentu saja, kita perlu mengekstrak tabel itu sendiri dari daftar Tabel dalam kasus ini.

```
>MT=Table[1];
```

Table is not a variable!
 Error in:
 MT=Table[1]; ...
 ^

Tentu saja, kita juga dapat menggunakan ini untuk menentukan nilai rata-rata kolom atau nilai statistik lainnya.

```
>mean(tablecol(MT,6))
```

Variable or function MT not found.
 Error in:
 mean(tablecol(MT,6)) ...
 ^

Fungsi `getstatistics()` mengembalikan elemen dalam vektor, dan jumlahnya. Kami menerapkannya pada nilai "m" dan "f" di kolom kedua tabel kami.

```
>{xu,count}=getstatistics(tablecol(MT,2)); xu, count,
```

Variable or function MT not found.

Error in:

```
{xu,count}=getstatistics(tablecol(MT,2)); xu, count, ...  
^
```

Kami dapat mencetak hasilnya dalam tabel baru.

```
>writetable(count',labr=tok[xu])
```

Variable count not found!

Error in:

```
writetable(count',labr=tok[xu]) ...  
^
```

Fungsi `selectable()` mengembalikan tabel baru dengan nilai dalam satu kolom yang dipilih dari vektor indeks. Pertama kita mencari indeks dari dua nilai kita di tabel token.

```
>v:=indexof(tok, ["g", "vg"])
```

Variable or function tok not found.

Error in:

```
v:=indexof(tok, ["g", "vg"]) ...  
^
```

Sekarang kita dapat memilih baris tabel, yang memiliki salah satu nilai dalam v di baris ke-5.

```
>MT1:=MT[selectrows(MT, 5, v)]; i:=sortedrows(MT1, 5);
```

Variable or function MT not found.

Error in:

```
MT1:=MT[selectrows(MT, 5, v)]; i:=sortedrows(MT1, 5); ...  
^
```

Sekarang kita dapat mencetak tabel, dengan nilai yang diekstrak dan diurutkan di kolom ke-5.

```
>writetable(MT1[i],labc=hd,ctok=ctok,tok=tok,wc=7);
```

Variable or function i not found.

Error in:

```
writetable(MT1[i],labc=hd,ctok=ctok,tok=tok,wc=7); ...  
^
```

Untuk statistik berikutnya, kami ingin menghubungkan dua kolom tabel. Jadi kami mengekstrak kolom 2 dan 4 dan mengurutkan tabel.

```
>i=sortedrows(MT,[2,4]); ...  
> writetable(tablecol(MT[i],[2,4])',ctok=[1,2],tok=tok)
```

Variable or function MT not found.

Error in:

```
i=sortedrows(MT,[2,4]);      writetable(tablecol(MT[i],[2,4])',c ...  
^
```

Dengan getstatistics(), kita juga dapat menghubungkan hitungan dalam dua kolom tabel satu sama lain.

```
>MT24=tablecol(MT,[2,4]); ...  
>{xu1,xu2,count}=getstatistics(MT24[1],MT24[2]); ...  
>writetable(count,labr=tok[xu1],labc=tok[xu2])
```

Variable or function MT not found.

Error in:

```
MT24=tablecol(MT,[2,4]); {xu1,xu2,count}=getstatistics(MT24[1] ...  
^
```

Sebuah tabel dapat ditulis ke file.

```
>filename="test.dat"; ...  
>writetable(count,labr=tok[xu1],labc=tok[xu2],file=filename);
```

Variable or function count not found.

Error in:

```
filename="test.dat"; writetable(count,labr=tok[xu1],labc=tok[x ...  
^
```

Kemudian kita bisa membaca tabel dari file.

```
>{MT2,hd,tok2,hdr}=readtable(filename,>clabs,>rlabs); ...
>writetable(MT2,labr=hdr,labc=hd)
```

```
Could not open the file
test.dat
for reading!
Try "trace errors" to inspect local variables after errors.
readtable:
    if filename!=none then open(filename,"r"); endif;
```

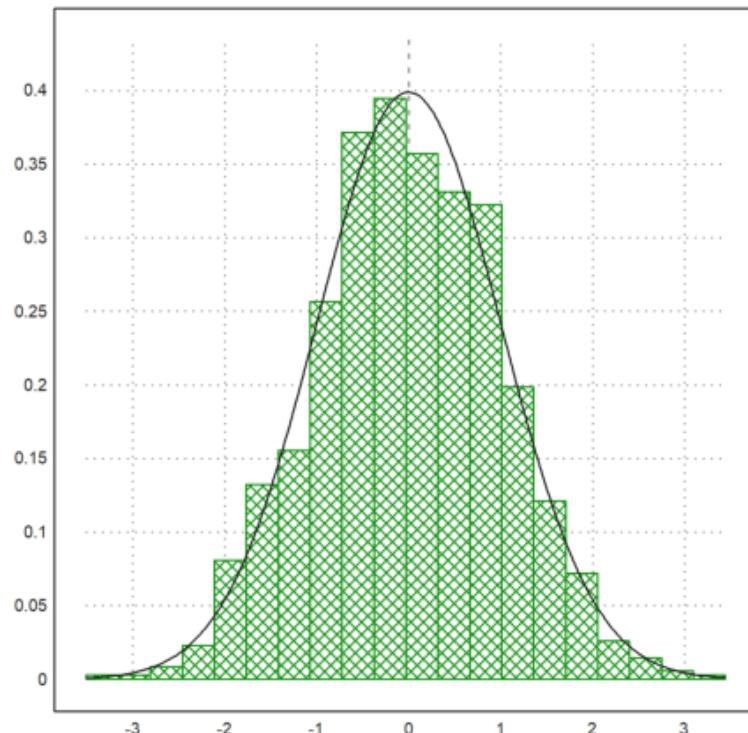
Dan hapus filenya.

```
>fileremove(filename);
```

Distribusi

Dengan plot2d, ada metode yang sangat mudah untuk memplot distribusi data eksperimen.

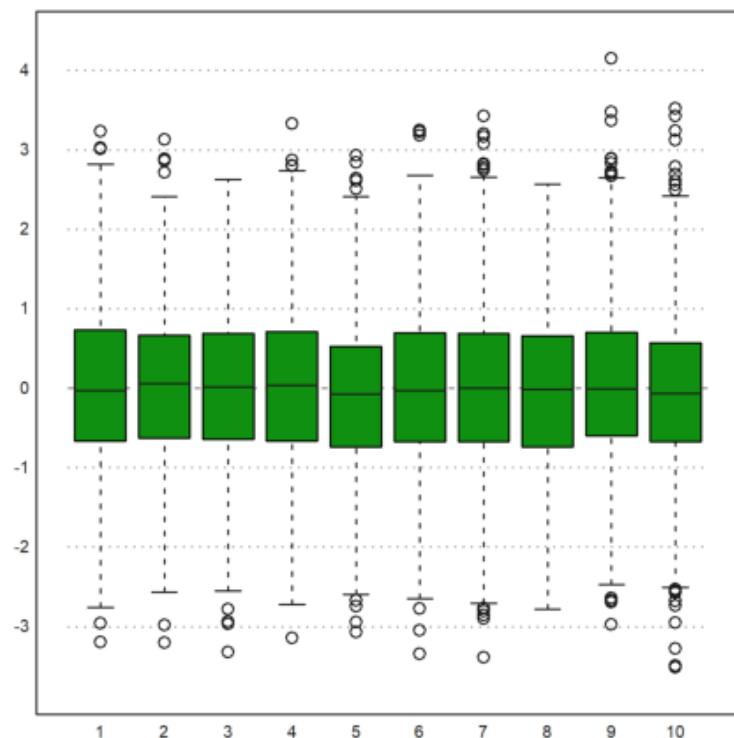
```
>p=normal(1,1000); //1000 random normal-distributed sample p
>plot2d(p,distribution=20,style="\/"); // plot the random sample p
>plot2d("qnormal(x,0,1)",add=1); // add the standard normal distribution pl
```



Harap dicatat perbedaan antara plot batang (sampel) dan kurva normal (distribusi nyata). Masukkan kembali tiga perintah untuk melihat hasil pengambilan sampel lainnya.

Berikut adalah perbandingan 10 simulasi 1000 nilai terdistribusi normal menggunakan apa yang disebut plot kotak. Plot ini menunjukkan median, kuartil 25% dan 75%, nilai minimal dan maksimal, dan outlier.

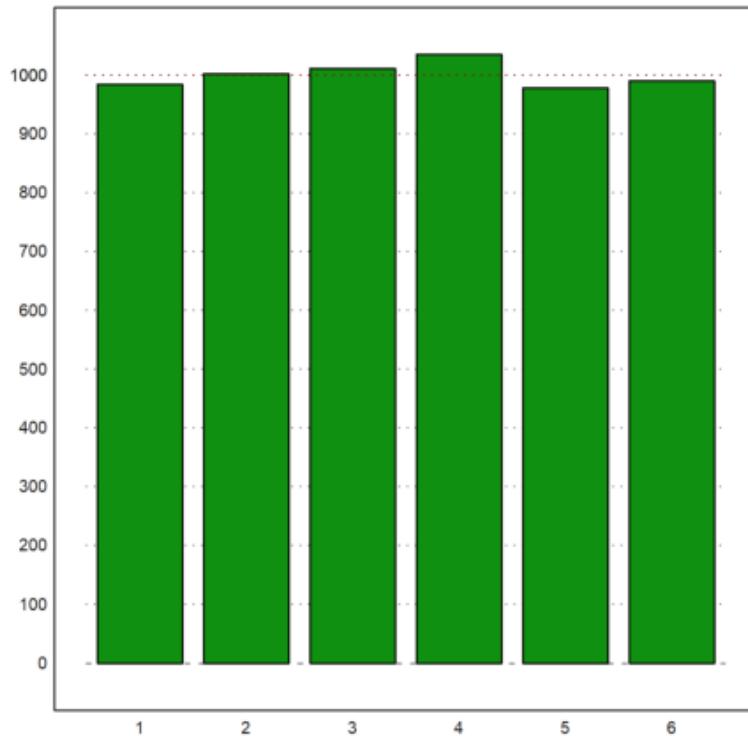
```
>p=normal(10,1000); boxplot(p):
```



Untuk menghasilkan bilangan bulat acak, Euler memiliki intrarandom. Mari kita simulasi lemparan dadu dan plot distribusinya.

Kami menggunakan fungsi getmultiplicities(v,x), yang menghitung seberapa sering elemen v muncul di x. Kemudian kita plot hasilnya menggunakan columnplot().

```
>k=intrandom(1,6000,6); ...
>columnplot(getmultiplicities(1:6,k)); ...
>ygrid(1000,color=red):
```



Sementara intrandom(n,m,k) mengembalikan bilangan bulat terdistribusi seragam dari 1 ke k, dimungkinkan untuk menggunakan distribusi bilangan bulat lainnya dengan randpint(). Dalam contoh berikut, probabilitas untuk 1,2,3 berturut-turut adalah 0,4,0,1,0,5.

```
>randpint(1,1000,[0.4,0.1,0.5]); getmultiplicities(1:3,%)
```

```
[378, 102, 520]
```

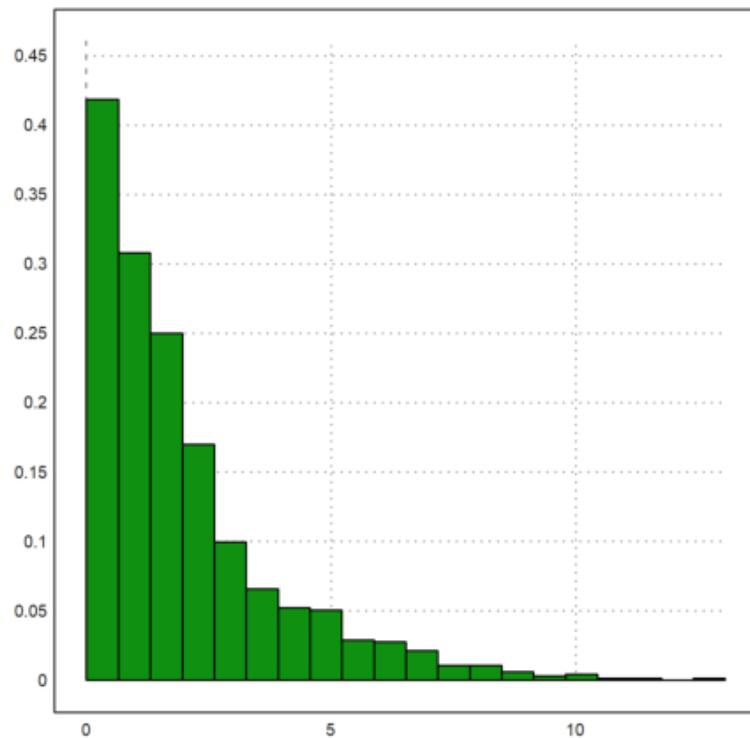
Euler dapat menghasilkan nilai acak dari lebih banyak distribusi. Coba lihat referensinya. Misalnya, kami mencoba distribusi eksponensial. Sebuah variabel acak kontinu X dikatakan memiliki distribusi eksponensial, jika PDF-nya diberikan oleh

$$f_X(x) = \lambda e^{-\lambda x}, \quad x > 0, \quad \lambda > 0,$$

with parameter

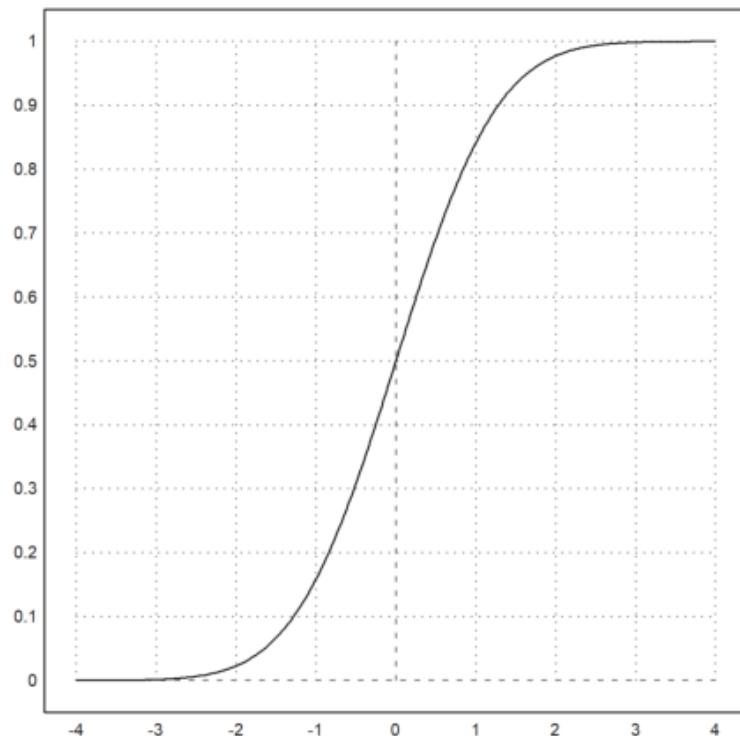
$$\lambda = \frac{1}{\mu}, \quad \mu \text{ is the mean, and denoted by } X \sim \text{Exponential}(\lambda).$$

```
>plot2d(randexponential(1,1000,2),>distribution):
```



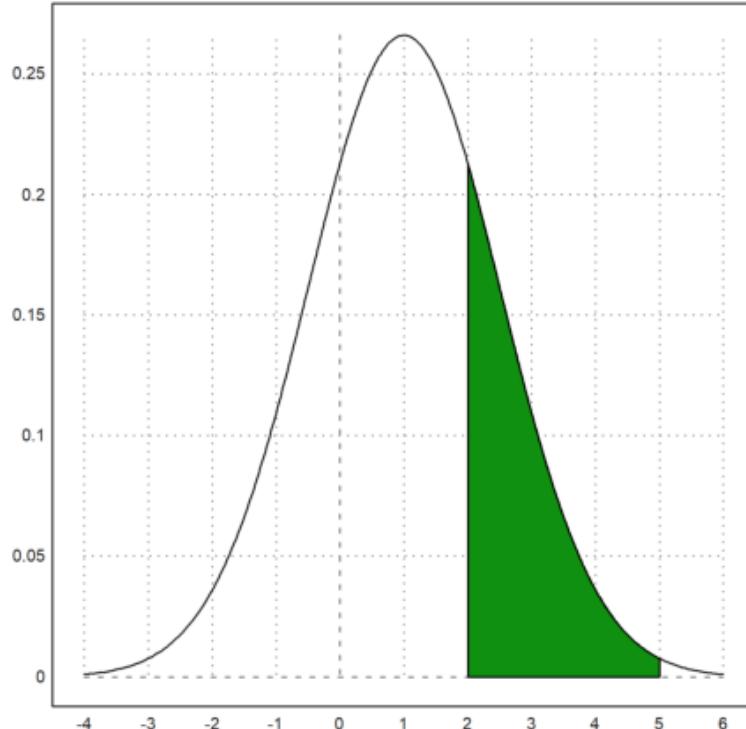
Untuk banyak distribusi, Euler dapat menghitung fungsi distribusi dan kebalikannya.

```
>plot2d("normaldis", -4, 4) :
```



Berikut ini adalah salah satu cara untuk memplot kuantil.

```
>plot2d("qnormal(x,1,1.5)",-4,6); ...
>plot2d("qnormal(x,1,1.5)",a=2,b=5,>add,>filled):
```



$$\text{normaldis}(x,m,d) = \int_{-\infty}^x \frac{1}{d\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{t-m}{d})^2} dt.$$

Peluang berada di area hijau adalah sebagai berikut.

```
>normaldis(5,1,1.5)-normaldis(2,1,1.5)
```

0.248662156979

Ini dapat dihitung secara numerik dengan integral berikut.

$$\int_2^5 \frac{1}{1.5\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-1}{1.5})^2} dx.$$

```
>gauss("qnormal(x,1,1.5)",2,5)
```

0.248662156979

Mari kita bandingkan distribusi binomial dengan distribusi normal mean dan deviasi yang sama. Fungsi invbindis() memecahkan interpolasi linier antara nilai integer.

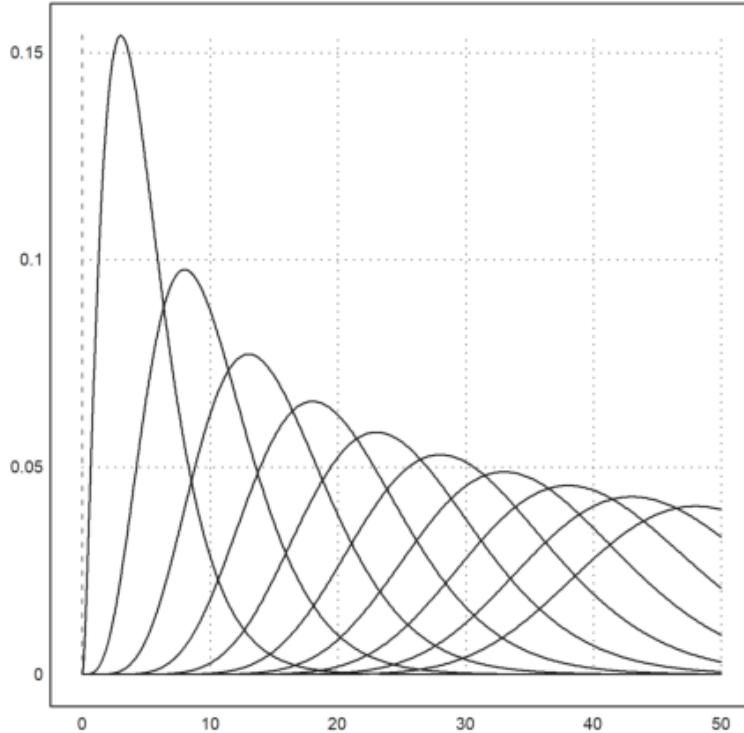
```
>invbindis(0.95,1000,0.5), invnormaldis(0.95,500,0.5*sqrt(1000))
```

525.516721219

526.007419394

Fungsi qdis() adalah densitas dari distribusi chi-kuadrat. Seperti biasa, evolusi vektor ke fungsi ini. Dengan demikian kita mendapatkan plot semua distribusi chi-kuadrat dengan derajat 5 sampai 30 dengan mudah dengan cara berikut.

```
>plot2d("qchidis(x, (5:5:50)')", 0, 50):
```



Euler memiliki fungsi yang akurat untuk mengevaluasi distribusi. Mari kita periksa chidis() dengan integral.

Penamaan mencoba untuk konsisten. Misalnya.,

- distribusi chi-kuadrat adalah chidis(),
- fungsi kebalikannya adalah invchidis(),
- kepadatannya adalah qchidis().

Komplemen dari distribusi (ekor atas) adalah chicdis().

```
>chidis(1.5,2), integrate("qchidis(x,2)",0,1.5)
```

0.527633447259
0.527633447259

Distribusi Diskrit

Untuk menentukan distribusi diskrit Anda sendiri, Anda dapat menggunakan metode berikut.

Pertama kita atur fungsi distribusinya.

```
>wd = 0 | ((1:6)+[-0.01,0.01,0,0,0,0])/6
```

[0, 0.165, 0.335, 0.5, 0.666667, 0.833333, 1]

Artinya dengan probabilitas $wd[i+1]-wd[i]$ kita menghasilkan nilai acak i.

Ini adalah distribusi yang hampir seragam. Mari kita mendefinisikan generator nomor acak untuk ini. Fungsi $find(v,x)$ menemukan nilai x dalam vektor v. Fungsi ini juga berfungsi untuk vektor x.

```
>function wrongdice (n,m) := find(wd,random(n,m))
```

Kesalahannya sangat halus sehingga kita hanya melihatnya dengan sangat banyak iterasi.

```
>columnsplot(getmultiplicities(1:6,wrongdice(1,1000000)):
```

images/EMT4Statistika_Saphira-011.png

Berikut adalah fungsi sederhana untuk memeriksa distribusi seragam dari nilai 1...K dalam v. Kami menerima hasilnya, jika untuk semua frekuensi

$$\left| f_i - \frac{1}{K} \right| < \frac{\delta}{\sqrt{n}}.$$

```
>function checkrandom (v, delta=1) ...
```

```
K=max(v); n=cols(v);  
fr=getfrequencies(v,1:K);  
return max(fr/n-1/K)<delta/sqrt(n);  
endfunction
```

Memang fungsi menolak distribusi seragam.

```
>checkrandom(wrongdice(1,1000000))
```

0

Dan itu menerima generator acak bawaan.

```
>checkrandom(intrandom(1,1000000,6))
```

1

Kita dapat menghitung distribusi binomial. Pertama ada `binomials()`, yang mengembalikan probabilitas i atau kurang hit dari n percobaan.

```
>bindis(410,1000,0.4)
```

0.751401349654

Fungsi Beta terbalik digunakan untuk menghitung interval kepercayaan Clopper-Pearson untuk parameter p. Tingkat default adalah alfa.

Arti interval ini adalah jika p berada di luar interval, hasil pengamatan 410 dalam 1000 jarang terjadi.

```
>clopperpearson(410,1000)
```

[0.37932, 0.441212]

Perintah berikut adalah cara langsung untuk mendapatkan hasil di atas. Tetapi untuk n besar, penjumlahan langsung tidak akurat dan lambat.

```
>p=0.4; i=0:410; n=1000; sum(bin(n,i)*p^i*(1-p)^(n-i))
```

0.751401349655

Omong-omong, `invbinsum()` menghitung kebalikan dari `binomials()`.

```
>invbindis(0.75,1000,0.4)
```

409.932733047

Di Bridge, kami mengasumsikan 5 kartu yang beredar (dari 52) di dua tangan (26 kartu). Mari kita hitung probabilitas distribusi yang lebih buruk dari 3:2 (misalnya 0:5, 1:4, 4:1 atau 5:0).

```
>2*hypergeomsum(1,5,13,26)
```

0.321739130435

Ada juga simulasi distribusi multinomial.

```
>randmultinomial(10,1000,[0.4,0.1,0.5])
```

381	100	519
376	91	533
417	80	503
440	94	466
406	112	482
408	94	498
395	107	498
399	96	505
428	87	485
400	99	501

Merencanakan Data

Untuk plot data, kami mencoba hasil pemilu Jerman sejak tahun 1990, diukur dalam kursi.

```
>BW := [ ...
>1990,662,319,239,79,8,17; ...
>1994,672,294,252,47,49,30; ...
>1998,669,245,298,43,47,36; ...
>2002,603,248,251,47,55,2; ...
>2005,614,226,222,61,51,54; ...
>2009,622,239,146,93,68,76; ...
>2013,631,311,193,0,63,64];
```

Untuk para pihak, kami menggunakan serangkaian nama.

```
>P := ["CDU/CSU", "SPD", "FDP", "Gr", "Li"];
```

Mari kita mencetak persentase dengan baik.

Pertama kita ekstrak kolom yang diperlukan. Kolom 3 sampai 7 adalah kursi masing-masing partai, dan kolom 2 adalah jumlah kursi. kolom 1 adalah tahun pemilihan.

```
>BT:=BW[,3:7]; BT:=BT/sum(BT); YT:=BW[,1]';
```

Kemudian kami mencetak statistik dalam bentuk tabel. Kami menggunakan nama sebagai tajuk kolom, dan tahun sebagai tajuk untuk baris. Lebar default untuk kolom adalah `wc=10`, tetapi kami lebih memilih output yang lebih padat. Kolom akan diperluas untuk label kolom, jika perlu.

```
>writetable(BT*100,wc=6,dc=0,>fixed,labc=P,labr=YT)
```

	CDU/CSU	SPD	FDP	Gr	Li
1990	48	36	12	1	3
1994	44	38	7	7	4
1998	37	45	6	7	5
2002	41	42	8	9	0
2005	37	36	10	8	9
2009	38	23	15	11	12
2013	49	31	0	10	10

Perkalian matriks berikut mengekstrak jumlah persentase dua partai besar yang menunjukkan bahwa partai-partai kecil telah memperoleh rekaman di parlemen hingga 2009.

```
>BT1:=(BT.[1;1;0;0;0])'*100
```

```
[84.29, 81.25, 81.1659, 82.7529, 72.9642, 61.8971, 79.8732]
```

Ada juga plot statistik sederhana. Kami menggunakannya untuk menampilkan garis dan titik secara bersamaan. Alternatifnya adalah memanggil `plot2d` dua kali dengan `>add`.

```
>statplot(YT,BT1,"b"):
```



images/EMT4Statistika_Saphira-012.png

Tentukan beberapa warna untuk masing-masing pihak.

```
>CP:=[rgb(0.5,0.5,0.5),red,yellow,green,rgb(0.8,0,0)];
```

Sekarang kita dapat memplot hasil pemilu 2009 dan perubahannya menjadi satu plot menggunakan gambar. Kita dapat menambahkan vektor kolom ke setiap plot.

```
>figure(2,1); ...
>figure(1); columnsplot(BW[6,3:7],P,color=CP); ...
>figure(2); columnsplot(BW[6,3:7]-BW[5,3:7],P,color=CP); ...
>figure(0):
```



images/EMT4Statistika_Saphira-013.png

Plot data menggabungkan deretan data statistik dalam satu plot.

```
>J:=BW[,1]'; DP:=BW[,3:7]'; ...
>dataplot(YT,BT',color=CP); ...
>labelbox(P,colors=CP,styles="[]",>points,w=0.2,x=0.3,y=0.4):
```



images/EMT4Statistika_Saphira-014.png

Sebuah kolom plot 3D menunjukkan baris data statistik dalam bentuk kolom. Kami menyediakan label untuk baris dan kolom. angle adalah sudut pandang.

```
>columnspplot3d(BT, scols=P, srows=YT, ...
>  angle=30°, ccols=CP) :
```



images/EMT4Statistika_Saphira-015.png

Representasi lain adalah plot mosaik. Perhatikan bahwa kolom plot mewakili kolom matriks di sini. Karena panjangnya label CDU/CSU, kami mengambil jendela yang lebih kecil dari biasanya.

```
>shrinkwindow(>smaller);  ...
>mosaicplot(BT', srows=YT, scols=P, color=CP, style="#"); ...
>shrinkwindow():
```



images/EMT4Statistika_Saphira-016.png

Kita juga bisa membuat diagram lingkaran. Karena hitam dan kuning membentuk koalisi, kami menyusun ulang elemen-elemennya.

```
>i=[1,3,5,4,2]; piechart(BW[6,3:7][i],color=CP[i],lab=P[i]):
```



images/EMT4Statistika_Saphira-017.png

Berikut adalah jenis plot lainnya.

```
>starplot(normal(1,10)+4,lab=1:10,>rays) :
```



images/EMT4Statistika_Saphira-018.png

Beberapa plot di plot2d bagus untuk statika. Berikut adalah plot impuls dari data acak, terdistribusi secara merata di [0,1].

```
>plot2d(makeimpulse(1:10,random(1,10)),>bar) :
```

A large rectangular box with a thin green border, occupying most of the page below the header. It contains the text "images/EMT4Statistika_Saphira-019.png".

images/EMT4Statistika_Saphira-019.png

Tetapi untuk data yang terdistribusi secara eksponensial, kita mungkin memerlukan plot logaritmik.

```
>logimpulseplot(1:10,-log(random(1,10))*10):
```

A large rectangular box with a thin green border, occupying most of the page below the code. It contains the text "images/EMT4Statistika_Saphira-020.png".

images/EMT4Statistika_Saphira-020.png

Fungsi `columnplot()` lebih mudah digunakan, karena hanya membutuhkan vektor nilai. Selain itu, ia dapat mengatur labelnya ke apa pun yang kita inginkan, kita sudah mendemonstrasikannya dalam tutorial ini.

Ini adalah aplikasi lain, di mana kita menghitung karakter dalam sebuah kalimat dan menyusun statistik.

```
>v=strtochar("the quick brown fox jumps over the lazy dog"); ...
>w=ascii("a"):ascii("z"); x=getmultiplicities(w,v); ...
>cw=[]; for k=w; cw=cw|char(k); end; ...
>columnsplot(x,lab=cw,width=0.05):
```



images/EMT4Statistika_Saphira-021.png

Dimungkinkan juga untuk mengatur sumbu secara manual.

```
>n=10; p=0.4; i=0:n; x=bin(n,i)*p^i*(1-p)^(n-i); ...
>columnsplot(x,lab=i,width=0.05,<frame,<grid); ...
>yaxis(0,0:0.1:1,style="->",>left); xaxis(0,style="."); ...
>label("p",0,0.25), label("i",11,0); ...
>textbox(["Binomial distribution","with p=0.4"]):
```



images/EMT4Statistika_Saphira-022.png

Berikut ini adalah cara untuk memplot frekuensi bilangan dalam sebuah vektor. Kami membuat vektor bilangan bulat bilangan acak 1 hingga 6.

```
>v:=intrandom(1,10,10)
```

```
[8, 5, 8, 8, 6, 8, 8, 3, 5, 5]
```

Kemudian ekstrak nomor unik di v.

```
>vu:=unique(v)
```

```
[3, 5, 6, 8]
```

Dan plot frekuensi dalam plot kolom.

```
>columnsplot(getmultiplicities(vu,v),lab=vu,style="/"):
```



images/EMT4Statistika_Saphira-023.png

Kami ingin menunjukkan fungsi untuk distribusi nilai empiris.

```
>x=normal(1,20);
```

Fungsi empdist(x,vs) membutuhkan array nilai yang diurutkan. Jadi kita harus mengurutkan x sebelum kita dapat menggunakannya.

```
>xs=sort(x);
```

Kemudian kami memplot distribusi empiris dan beberapa batang kepadatan menjadi satu plot. Alih-alih plot bar untuk distribusi, kami menggunakan plot gigi gergaji kali ini.

```
>figure(2,1); ...
>figure(1); plot2d("empdist",-4,4;xs); ...
>figure(2); plot2d(histo(x,v=-4:0.2:4,<bar>)); ...
>figure(0):
```



images/EMT4Statistika_Saphira-024.png

Plot pencar mudah dilakukan di Euler dengan plot titik biasa. Grafik berikut menunjukkan bahwa X dan $X+Y$ jelas berkorelasi positif.

```
>x=normal(1,100); plot2d(x,x+rotright(x),>points,style=".."):
```



images/EMT4Statistika_Saphira-025.png

Seringkali, kita ingin membandingkan dua sampel dari distribusi yang berbeda. Ini dapat dilakukan dengan plot kuantil-kuantil.

Untuk pengujian, kami mencoba distribusi student-t dan distribusi eksponensial.

```
>x=randt(1,1000,5); y=randnormal(1,1000,mean(x),dev(x)); ...
>plot2d("x",r=6,style="--",yl="normal",xl="student-t",>vertical); ...
>plot2d(sort(x),sort(y),>points,color=red,style="x",>add):
```



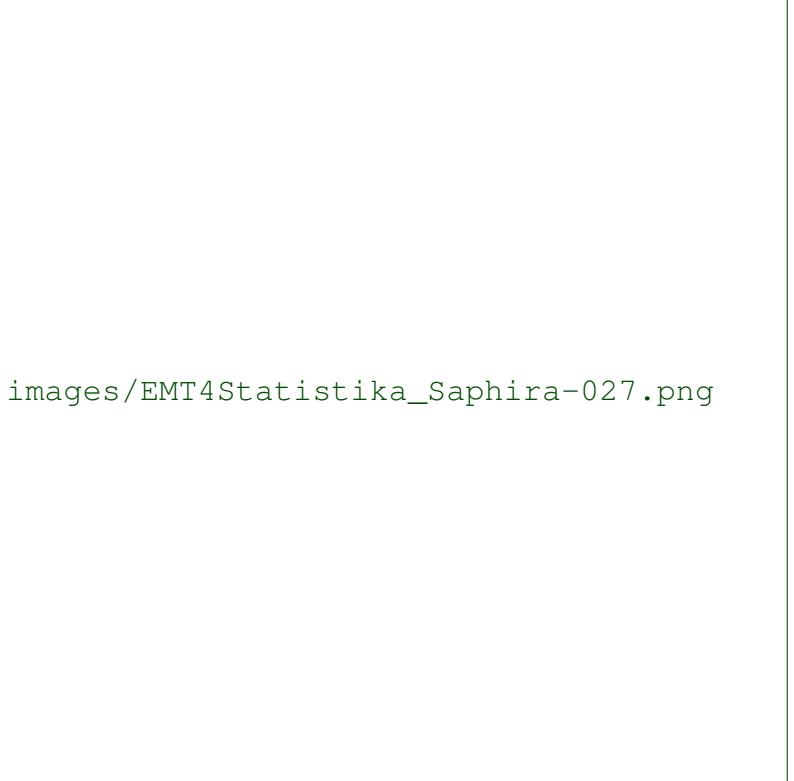
Plot dengan jelas menunjukkan bahwa nilai terdistribusi normal cenderung lebih kecil di ujung ekstrim.

Jika kita memiliki dua distribusi dengan ukuran yang berbeda, kita dapat memperluas yang lebih kecil atau mengecilkan yang lebih besar. Fungsi berikut baik untuk keduanya. Dibutuhkan nilai median dengan persentase antara 0 dan 1.

```
>function medianexpand (x,n) := median(x,p=linspace(0,1,n-1));
```

Mari kita bandingkan dua distribusi yang sama.

```
>x=random(1000); y=random(400); ...
>plot2d("x",0,1,style="--"); ...
>plot2d(sort(medianexpand(x,400)),sort(y),>points,color=red,style="x",>add)
```

A large rectangular box with a thin green border, positioned in the upper right quadrant of the page. It appears to be a placeholder for a figure or image.

images/EMT4Statistika_Saphira-027.png

Regresi dan Korelasi

Regresi linier dapat dilakukan dengan fungsi polyfit() atau berbagai fungsi fit.

Sebagai permulaan, kami menemukan garis regresi untuk data univariat dengan polifit(x,y,1).

```
>x=1:10; y=[2,3,1,5,6,3,7,8,9,8]; writetable(x' | y', labc= ["x", "y"] )
```

x	y
1	2
2	3
3	1
4	5
5	6
6	3
7	7
8	8
9	9
10	8

Kami ingin membandingkan non-weighted dan weighted fit. Pertama, koefisien kecocokan linier.

```
>p=polyfit(x,y,1)

[0.733333, 0.812121]
```

Sekarang koefisien dengan bobot yang menekankan nilai terakhir.

```
>w &= "exp(-(x-10)^2/10)"; pw=polyfit(x,y,1,w=w(x))

[4.71566, 0.38319]
```

Kami memasukkan semuanya ke dalam satu plot untuk titik dan garis regresi, dan untuk bobot yang digunakan.

```
>figure(2,1); ...
>figure(1); statplot(x,y,"b",xl="Regression"); ...
> plot2d("evalpoly(x,p)",>add,color=blue,style="--"); ...
> plot2d("evalpoly(x,pw)",5,10,>add,color=red,style="--"); ...
>figure(2); plot2d(w,1,10,>filled,style="/",fillcolor=red,xl=w); ...
>figure(0):
```

images/EMT4Statistika_Saphira-028.png

Sebagai contoh lain kita membaca survei siswa, usia mereka, usia orang tua mereka dan jumlah saudara kandung dari sebuah file.

Tabel ini berisi "m" dan "f" di kolom kedua. Kami menggunakan variabel tok2 untuk mengatur terjemahan yang tepat daripada membiarkan readtable() mengumpulkan terjemahan.

```
>{MS,hd}:=readtable("table1.dat",tok2:=[ "m", "f" ]); ...  
>writetable(MS,labc=hd,tok2:=[ "m", "f" ]);
```

```
Could not open the file  
table1.dat  
for reading!  
Try "trace errors" to inspect local variables after errors.  
readtable:  
    if filename!=none then open(filename,"r"); endif;
```

Bagaimana usia bergantung satu sama lain? Kesan pertama datang dari scatterplot berpasangan.

```
>scatterplots(tablecol(MS, 3:5),hd[3:5]):
```

```
Variable or function MS not found.  
Error in:  
scatterplots(tablecol(MS, 3:5),hd[3:5]): ...  
^
```

Jelas bahwa usia ayah dan ibu bergantung satu sama lain. Mari kita tentukan dan plot garis regresinya.

```
>cs:=MS[, 4:5]'; ps:=polyfit(cs[1],cs[2],1)  
  
MS is not a variable!  
Error in:  
cs:=MS[, 4:5]'; ps:=polyfit(cs[1],cs[2],1) ...  
^
```

Ini jelas model yang salah. Garis regresinya adalah $s=17+0,74t$, di mana t adalah usia ibu dan s usia ayah. Perbedaan usia mungkin sedikit bergantung pada usia, tetapi tidak terlalu banyak.

Sebaliknya, kami menduga fungsi seperti $s=a+t$. Maka a adalah mean dari s-t. Ini adalah perbedaan usia rata-rata antara ayah dan ibu.

```
>da:=mean(cs[2]-cs[1])
```

```
cs is not a variable!
Error in:
da:=mean(cs[2]-cs[1]) ...  
^
```

Mari kita plot ini menjadi satu plot pencar.

```
>plot2d(cs[1],cs[2],>points); ...
>plot2d("evalpoly(x,ps)",color=red,style=".",>add); ...
>plot2d("x+da",color=blue,>add):
```

```
cs is not a variable!
Error in:
plot2d(cs[1],cs[2],>points); plot2d("evalpoly(x,ps)",color=re ...  
^
```

Berikut adalah plot kotak dari dua zaman. Ini hanya menunjukkan, bahwa usianya berbeda.

```
>boxplot(cs, ["mothers", "fathers"]):
```

```
Variable or function cs not found.
Error in:
boxplot(cs, ["mothers", "fathers"]): ...
^
```

Sangat menarik bahwa perbedaan median tidak sebesar perbedaan rata-rata.

```
>median(cs[2])-median(cs[1])
```

```
cs is not a variable!
Error in:
median(cs[2])-median(cs[1]) ...  
^
```

Koefisien korelasi menunjukkan korelasi positif.

```
>correl(cs[1],cs[2])
```

```
cs is not a variable!
Error in:
correl(cs[1],cs[2]) ...  
^
```

Korelasi peringkat adalah ukuran untuk urutan yang sama di kedua vektor. Ini juga cukup positif.

```
>rankcorrel(cs[1],cs[2])
```

```
cs is not a variable!
Error in:
rankcorrel(cs[1],cs[2]) ...
^
```

Membuat Fungsi baru

Tentu saja, bahasa EMT dapat digunakan untuk memprogram fungsi-fungsi baru. Misalnya, kita mendefinisikan fungsi skewness.

$$\text{sk}(x) = \frac{\sqrt{n} \sum_i (x_i - m)^3}{(\sum_i (x_i - m)^2)^{3/2}}$$

dimana m adalah mean dari x.

```
>function skew (x:vector) ...
m=mean(x);
return sqrt(cols(x))*sum((x-m)^3)/(sum((x-m)^2))^(3/2);
endfunction
```

Seperti yang Anda lihat, kita dapat dengan mudah menggunakan bahasa matriks untuk mendapatkan implementasi yang sangat singkat dan efisien. Mari kita coba fungsi ini.

```
>data=normal(20); skew(normal(10))
```

```
-0.198710316203
```

Berikut adalah fungsi lain, yang disebut koefisien skewness Pearson.

```
>function skew1 (x) := 3*(mean(x)-median(x))/dev(x)
>skew1(data)
```

```
-0.0801873249135
```

Simulasi Monte Carlo

Euler dapat digunakan untuk mensimulasikan kejadian acak. Kita telah melihat contoh sederhana di atas. Ini adalah satu lagi, yang mensimulasikan 1000 kali 3 lemparan dadu, dan meminta distribusi jumlah.

```
>ds := sum(intrandom(1000, 3, 6))'; fs = getmultiplicities(3:18, ds)
```

```
[5, 17, 35, 44, 75, 97, 114, 116, 143, 116, 104, 53, 40,
22, 13, 6]
```

kita bisa membuat plot ini sekarang

```
>columnsplot(fs, lab=3:18):
```



images/EMT4Statistika_Saphira-029.png

Untuk menentukan distribusi yang diharapkan tidak begitu mudah. Kami menggunakan rekursi lanjutan untuk ini.

Fungsi berikut menghitung banyaknya cara bilangan k dapat direpresentasikan sebagai jumlah n bilangan dalam rentang 1 sampai m. Ia bekerja secara rekursif dengan cara yang jelas.

```
>function map countways (k; n, m) ...
```

```

if n==1 then return k>=1 && k<=m
else
    sum=0;
    loop 1 to m; sum=sum+countways(k-#,n-1,m); end;
    return sum;
end;
endfunction

```

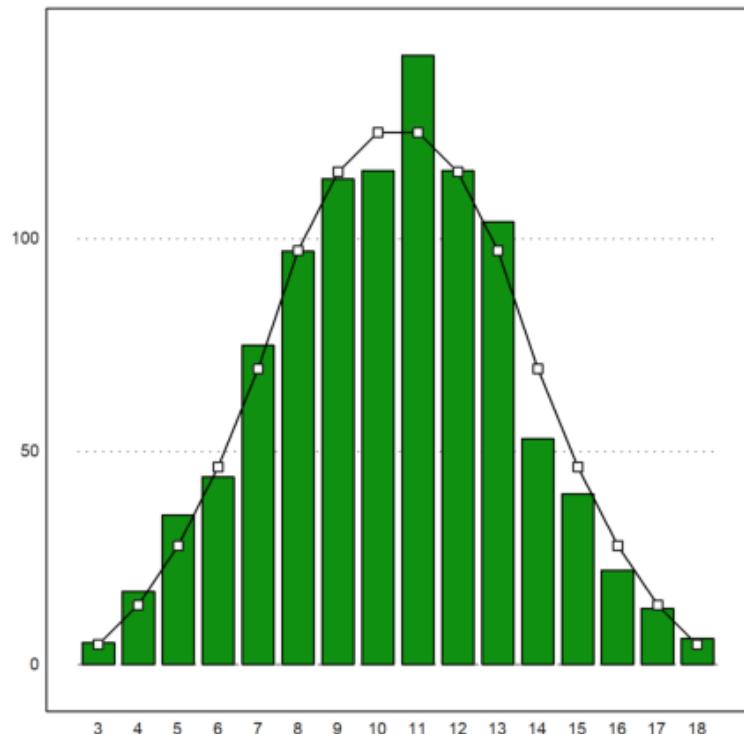
Berikut adalah hasil dari tiga lemparan dadu.

```
>cw=countways(3:18,3,6)
```

```
[1, 3, 6, 10, 15, 21, 25, 27, 27, 25, 21, 15, 10, 6, 3,
1]
```

Kami menambahkan nilai yang diharapkan ke plot.

```
>plot2d(cw/6^3*1000,>add); plot2d(cw/6^3*1000,>points,>add):
```



Untuk simulasi lain, simpangan nilai rata-rata dari n 0-1-variabel acak terdistribusi normal adalah $1/\sqrt{n}$.

```
>longformat; 1/sqrt(10)
```

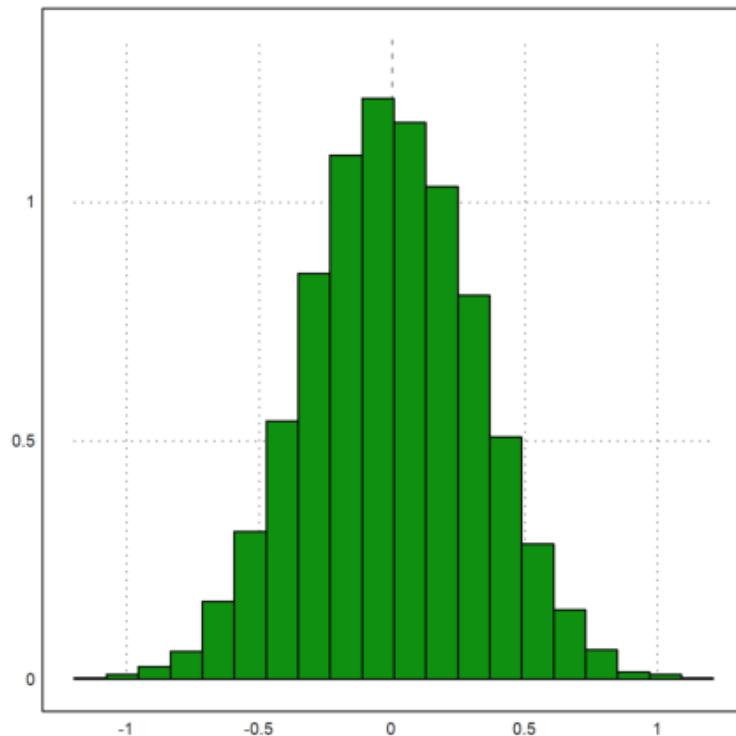
0.316227766017

Mari kita periksa ini dengan simulasi. Kami memproduksi 10000 kali 10 vektor acak.

```
>M=normal(10000,10); dev(mean(M)')
```

0.319493614817

```
>plot2d(mean(M)',>distribution):
```



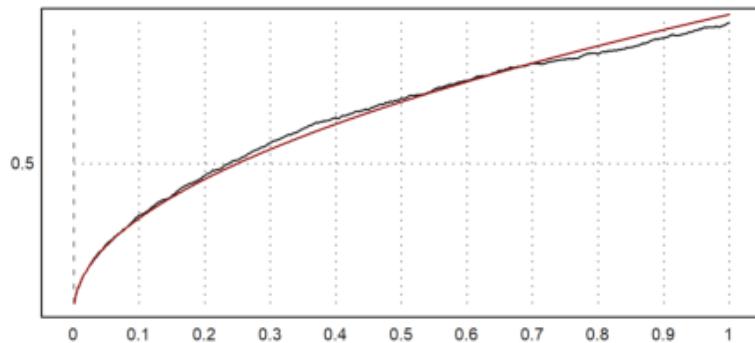
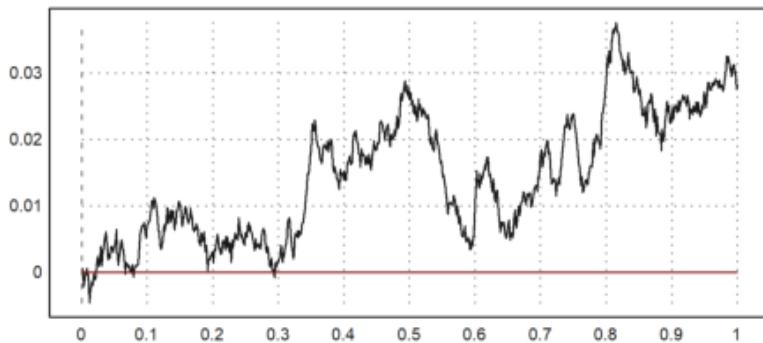
Median 10 0-1-bilangan acak terdistribusi normal memiliki simpangan yang lebih besar.

```
>dev(median(M)')
```

0.374460271535

Karena kita dapat dengan mudah menghasilkan jalan acak, kita dapat mensimulasikan proses Wiener. Kami mengambil 1000 langkah dari 1000 proses. Kami kemudian memplot deviasi standar dan rata-rata dari langkah ke-n dari proses ini bersama dengan nilai yang diharapkan dalam warna merah.

```
>n=1000; m=1000; M=cumsum(normal(n,m) / sqrt(m)) ; ...
>t=(1:n)/n; figure(2,1); ...
>figure(1); plot2d(t,mean(M'))'; plot2d(t,0,color=red,>add); ...
>figure(2); plot2d(t,dev(M'))'; plot2d(t,sqrt(t),color=red,>add); ...
>figure(0):
```



Uji

Uji adalah alat penting dalam statistik. Di Euler, banyak tes diimplementasikan. Semua tes ini mengembalikan kesalahan yang kami terima jika kami menolak hipotesis nol.

Sebagai contoh, kami menguji lemparan dadu untuk distribusi seragam. Pada 600 lemparan, kami mendapatkan nilai berikut, yang kami masukkan ke dalam uji chi-kuadrat.

```
>chitest([90,103,114,101,103,89],dup(100,6)')
```

0.498830517952

Tes chi-kuadrat juga memiliki mode, yang menggunakan simulasi Monte Carlo untuk mengejuti statistik. Hasilnya harus hampir sama. Parameter `>p` menginterpretasikan vektor-y sebagai vektor probabilitas.

```
>chitest ([90,103,114,101,103,89],dup (1/6,6)',>p,>montecarlo)
```

0.526

Kesalahan ini terlalu besar. Jadi kita tidak bisa menolak distribusi seragam. Ini tidak membuktikan bahwa dadu kami adil. Tapi kita tidak bisa menolak hipotesis kita.

Selanjutnya kita menghasilkan 1000 lemparan dadu menggunakan generator angka acak, dan melakukan tes yang sama.

```
>n=1000; t=random ([1,n*6]); chitest (count (t*6,6),dup (n,6)')
```

0.528028118442

Mari kita uji nilai rata-rata 100 dengan uji-t.

```
>s=200+normal ([1,100])*10; ...
>ttest (mean (s), dev (s), 100, 200)
```

0.0218365848476

Fungsi `ttest()` membutuhkan nilai rata-rata, simpangan, jumlah data, dan nilai rata-rata yang akan diuji.

Sekarang mari kita periksa dua pengukuran untuk mean yang sama. Kami menolak hipotesis bahwa mereka memiliki rata-rata yang sama, jika hasilnya $<0,05$.

```
>tcomparedata (normal (1,10),normal (1,10))
```

0.38722000942

Jika kita menambahkan bias ke satu distribusi, kita mendapatkan lebih banyak penolakan. Ulangi simulasi ini beberapa kali untuk melihat efeknya.

```
>tcomparedata (normal (1,10),normal (1,10)+2)
```

5.60009101758e-07

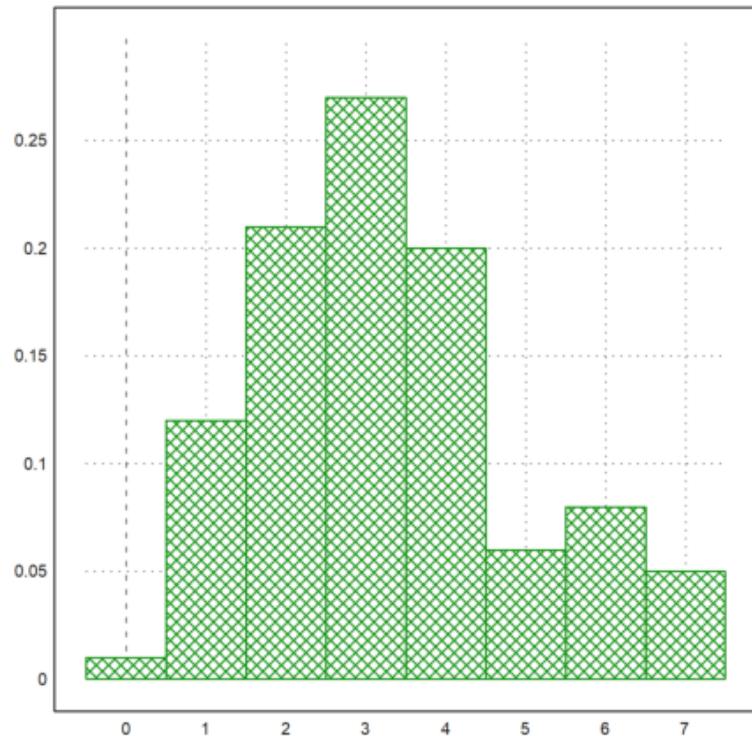
Pada contoh berikutnya, kita menghasilkan 20 lemparan dadu acak sebanyak 100 kali dan menghitung yang ada di dalamnya. Harus ada $20/6=3,3$ yang rata-rata.

```
>R=random(100,20); R=sum(R*6<=1)'; mean(R)
```

3.28

Kami sekarang membandingkan jumlah satu dengan distribusi binomial. Pertama kita plot distribusi yang.

```
>plot2d(R,distribution=max(R)+1,even=1,style="\/"):
```



```
>t=count(R,21);
```

Kemudian kami menghitung nilai yang diharapkan.

```
>n=0:20; b=bin(20,n)*(1/6)^n*(5/6)^(20-n)*100;
```

Kita harus mengumpulkan beberapa angka untuk mendapatkan kategori yang cukup besar.

```
>t1=sum(t[1:2])|t[3:7]|sum(t[8:21]); ...
>b1=sum(b[1:2])|b[3:7]|sum(b[8:21]);
```

Uji chi-kuadrat menolak hipotesis bahwa distribusi kami adalah distribusi binomial, jika hasilnya <0,05.

```
>chitest(t1,b1)
```

0.53921579764

Contoh berikut berisi hasil dua kelompok orang (laki-laki dan perempuan, katakanlah) memberikan suara untuk satu dari enam partai.

```
>A=[23,37,43,52,64,74;27,39,41,49,63,76]; ...
> writetable(A,wc=6,labr=["m","f"],labc=1:6)
```

	1	2	3	4	5	6
m	23	37	43	52	64	74
f	27	39	41	49	63	76

Kami ingin menguji independensi suara dari jenis kelamin. Tes tabel chi^2 melakukan ini. Akibatnya terlalu besar untuk menolak kemerdekaan. Jadi kita tidak bisa mengatakan, jika voting tergantung pada jenis kelamin dari data ini.

```
>tabletest(A)
```

0.990701632326

Berikut ini adalah tabel yang diharapkan, jika kita mengasumsikan frekuensi pemungutan suara yang diamati.

```
>writetable(expectedtable(A),wc=6,dc=1,labr=["m","f"],labc=1:6)
```

	1	2	3	4	5	6
m	24.9	37.9	41.9	50.3	63.3	74.7
f	25.1	38.1	42.1	50.7	63.7	75.3

Kita dapat menghitung koefisien kontingensi yang dikoreksi. Karena sangat dekat dengan 0, kami menyimpulkan bahwa pemungutan suara tidak tergantung pada jenis kelamin.

```
>contingency (A)
```

0.0427225484717

Uji Lainnya

Selanjutnya kami menggunakan analisis varians (Uji-F) untuk menguji tiga sampel data yang terdistribusi normal untuk nilai rata-rata yang sama. Metode tersebut disebut ANOVA (analisis varians). Di Euler, fungsi varanalysis() digunakan.

```
>x1=[109,111,98,119,91,118,109,99,115,109,94]; mean(x1),
```

106.545454545

```
>x2=[120,124,115,139,114,110,113,120,117]; mean(x2),
```

119.111111111

```
>x3=[120,112,115,110,105,134,105,130,121,111]; mean(x3)
```

116.3

```
>varanalysis (x1,x2,x3)
```

0.0138048221371

Ini berarti, kami menolak hipotesis nilai rata-rata yang sama. Kami melakukan ini dengan probabilitas kesalahan 1,3%.

Ada juga uji median, yang menolak sampel data dengan distribusi rata-rata berbeda menguji median sampel bersatu.

```
>a=[56,66,68,49,61,53,45,58,54];
>b=[72,81,51,73,69,78,59,67,65,71,68,71];
>mediantest (a,b)
```

0.0241724220052

Tes lain tentang kesetaraan adalah tes peringkat. Ini jauh lebih tajam daripada tes median.

```
>ranktest(a,b)
```

0.00199969612469

Dalam contoh berikut, kedua distribusi memiliki mean yang sama.

```
>ranktest(random(1,100),random(1,50)*3-1)
```

0.129608141484

Sekarang mari kita coba mensimulasikan dua perlakuan a dan b yang diterapkan pada orang yang berbeda.

```
>a=[8.0,7.4,5.9,9.4,8.6,8.2,7.6,8.1,6.2,8.9];
>b=[6.8,7.1,6.8,8.3,7.9,7.2,7.4,6.8,6.8,8.1];
```

Tes signum memutuskan, jika a lebih baik dari b.

```
>signtest(a,b)
```

0.0546875

Ini terlalu banyak kesalahan. Kita tidak dapat menolak bahwa a sama baiknya dengan b. Tes Wilcoxon lebih tajam dari tes ini, tetapi bergantung pada nilai kuantitatif perbedaan.

```
>wilcoxon(a,b)
```

0.0296680599405

Mari kita coba dua tes lagi menggunakan seri yang dihasilkan.

```
>>wilcoxon(normal(1,20),normal(1,20)-1)
```

0.0068706451766

```
>>wilcoxon(normal(1,20),normal(1,20))
```

0.275145971064

Berikut ini adalah pengujian untuk pembangkit bilangan acak. Euler menggunakan generator yang sangat bagus, jadi kita tidak perlu mengharapkan masalah.

Pertama kita menghasilkan sepuluh juta angka acak di $[0,1]$.

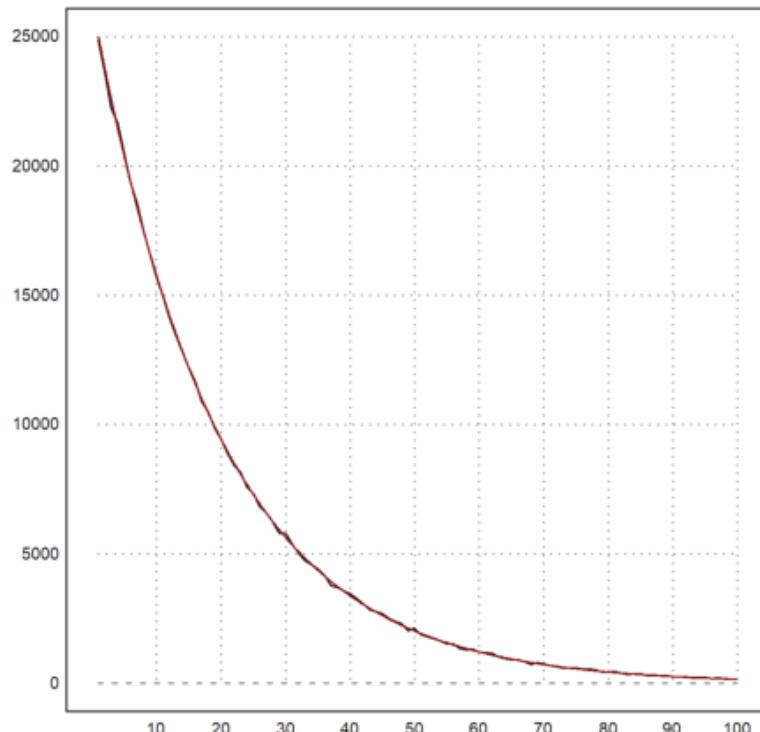
```
>n:=10000000; r:=random(1,n);
```

Selanjutnya kita hitung jarak antara dua bilangan kurang dari 0,05.

```
>a:=0.05; d:=differences(nonzeros(r<a));
```

Akhirnya, kami memplot berapa kali, setiap jarak terjadi, dan membandingkan dengan nilai yang diharapkan.

```
>m=getmultiplicities(1:100,d); plot2d(m); ...
> plot2d("n*(1-a)^(x-1)*a^2",color=red,>add):
```



Hapus datanya.

```
>remvalue n;
```

Pengantar untuk Pengguna Proyek R

Jelas, EMT tidak bersaing dengan R sebagai paket statistik. Namun, ada banyak prosedur dan fungsi statistik yang tersedia di EMT juga. Jadi EMT dapat memenuhi kebutuhan dasar. Bagaimanapun, EMT hadir dengan paket numerik dan sistem aljabar komputer.

Notebook ini cocok untuk Anda yang terbiasa dengan R, tetapi perlu mengetahui perbedaan sintaks EMT dan R. Kami mencoba memberikan gambaran tentang hal-hal yang jelas dan kurang jelas yang perlu Anda ketahui.

Selain itu, kami mencari cara untuk bertukar data antara kedua sistem.

Perhatikan bahwa ini adalah pekerjaan yang sedang berjalan.

Sintaks Dasar

Hal pertama yang Anda pelajari di R adalah membuat vektor. Di EMT, perbedaan utama adalah bahwa : operator dapat mengambil ukuran langkah. Selain itu, ia memiliki daya ikat yang rendah.

```
>n=10; 0:n/20:n-1
```

```
[0, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6, 6.5,
7, 7.5, 8, 8.5, 9]
```

Fungsi c() tidak ada. Dimungkinkan untuk menggunakan vektor untuk menggabungkan sesuatu.

Contoh berikut, seperti banyak contoh lainnya, dari "Introduction to R" yang disertakan dengan proyek R. Jika Anda membaca PDF ini, Anda akan menemukan bahwa saya mengikuti jalannya dalam tutorial ini.

```
>x=[10.4, 5.6, 3.1, 6.4, 21.7]; [x,0,x]
```

```
[10.4, 5.6, 3.1, 6.4, 21.7, 0, 10.4, 5.6, 3.1, 6.4, 21.7]
```

Operator titik dua dengan ukuran langkah EMT diganti dengan fungsi seq() di R. Kita bisa menulis fungsi ini di EMT.

```
>function seq(a,b,c) := a:b:c; ...
>seq(0,-0.1,-1)
```

```
[0, -0.1, -0.2, -0.3, -0.4, -0.5, -0.6, -0.7, -0.8, -0.9, -1]
```

Fungsi rep() dari R tidak ada di EMT. Untuk input vektor, dapat ditulis sebagai berikut.

```
>function rep(x:vector,n:index) := flatten(dup(x,n)); ...
>rep(x,2)
```

```
[10.4, 5.6, 3.1, 6.4, 21.7, 10.4, 5.6, 3.1, 6.4, 21.7]
```

Perhatikan bahwa "=" atau ":=" digunakan untuk tugas. Operator "->" digunakan untuk unit di EMT.

```
>125km -> " miles"
```

```
77.6713990297 miles
```

Operator "<->" untuk penugasan tetap menyesatkan, dan bukan ide yang baik untuk R. Berikut ini akan membandingkan a dan -4 di EMT.

```
>a=2; a<-4
```

```
0
```

Di R, "a<-4<3" berfungsi, tetapi "a<-4<-3" tidak. Saya juga memiliki ambiguitas serupa di EMT, tetapi mencoba menghilangkannya perlahan-lahan.

EMT dan R memiliki vektor bertipe boolean. Namun di EMT, angka 0 dan 1 digunakan untuk mewakili salah dan benar. Di R, nilai true dan false dapat digunakan dalam aritmatika biasa seperti di EMT.

```
>x<5, %*x
```

```
[0, 0, 1, 0, 0]
[0, 0, 3.1, 0, 0]
```

EMT melempar kesalahan atau menghasilkan NAN tergantung pada tanda "kesalahan".

```
>errors off; 0/0, isNaN(sqrt(-1)), errors on;
```

```
NAN
```

```
1
```

String sama di R dan EMT. Keduanya berada di lokal saat ini, bukan di Unicode.

Di R ada paket untuk Unicode. Di EMT, sebuah string dapat berupa string Unicode. String unicode dapat diterjemahkan ke pengkodean lokal dan sebaliknya. Selain itu, u"..." dapat berisi entitas HTML.

```
>u"    ; Ren   Grothmann"
```

© Ren   Grothmann

Berikut ini mungkin atau mungkin tidak ditampilkan dengan benar di sistem Anda sebagai A dengan titik dan garis di atasnya. Itu tergantung pada font yang Anda gunakan.

```
>chartoutf([480])
```

Penggabungan string dilakukan dengan "+" atau "|". Ini dapat mencakup angka, yang akan dicetak dalam format saat ini.

```
>"pi = "+pi
```

pi = 3.14159265359

Pengindeksan

Sebagian besar waktu, ini akan berfungsi seperti pada R.

Tetapi EMT akan menginterpretasikan indeks negatif dari belakang vektor, sedangkan R menginterpretasikan x[n] sebagai x tanpa elemen ke-n.

```
>x, x[1:3], x[-2]
```

```
[10.4, 5.6, 3.1, 6.4, 21.7]
[10.4, 5.6, 3.1]
6.4
```

Perilaku R dapat dicapai dalam EMT dengan drop().

```
>drop(x, 2)
```

```
[10.4, 3.1, 6.4, 21.7]
```

Vektor logis tidak diperlakukan secara berbeda sebagai indeks di EMT, berbeda dengan R. Anda perlu mengekstrak elemen bukan nol terlebih dahulu di EMT.

```
>x, x>5, x[nonzeros(x>5)]
```

```
[10.4, 5.6, 3.1, 6.4, 21.7]
[1, 1, 0, 1, 1]
[10.4, 5.6, 6.4, 21.7]
```

Sama seperti di R, vektor indeks dapat berisi pengulangan.

```
>x[[1,2,2,1]]
```

```
[10.4, 5.6, 5.6, 10.4]
```

Tetapi nama untuk indeks tidak dimungkinkan di EMT. Untuk paket statistik, ini mungkin sering diperlukan untuk memudahkan akses ke elemen vektor.

Untuk meniru perilaku ini, kita dapat mendefinisikan fungsi sebagai berikut.

```
>function sel (v,i,s) := v[indexof(s,i)]; ...
>s=["first","second","third","fourth"]; sel(x,[ "first","third"],s)
```

```
Trying to overwrite protected function sel!
Error in:
function sel (v,i,s) := v[indexof(s,i)]; ... ...
^
[10.4, 3.1]
```

Tipe Data

EMT memiliki lebih banyak tipe data tetap daripada R. Jelas, di R ada vektor yang tumbuh. Anda dapat mengatur vektor numerik kosong v dan menetapkan nilai ke elemen v[17]. Ini tidak mungkin di EMT.

Berikut ini agak tidak efisien.

```
>v=[]; for i=1 to 10000; v=v|i; end;
```

EMT sekarang akan membuat vektor dengan v dan i ditambahkan pada tumpukan dan menyalin vektor itu kembali ke variabel global v .

Semakin efisien pra-mendefinisikan vektor.

```
>v=zeros(10000); for i=1 to 10000; v[i]=i; end;
```

Untuk mengubah jenis tanggal di EMT, Anda dapat menggunakan fungsi seperti `complex()`.

```
>complex(1:4)
```

```
[ 1+0i , 2+0i , 3+0i , 4+0i ]
```

Konversi ke string hanya dimungkinkan untuk tipe data dasar. Format saat ini digunakan untuk rangkaian string sederhana. Tetapi ada fungsi seperti `print()` atau `frac()`.

Untuk vektor, Anda dapat dengan mudah menulis fungsi Anda sendiri.

```
>function tostr (v) ...
```

```
s="[";  
loop 1 to length(v);  
    s=s+print(v[#],2,0);  
    if #<length(v) then s=s+","; endif;  
end;  
return s+"]";  
endfunction
```

```
>tostr(linspace(0,1,10))
```

```
[0.00,0.10,0.20,0.30,0.40,0.50,0.60,0.70,0.80,0.90,1.00]
```

Untuk komunikasi dengan Maxima, terdapat fungsi `convertmxm()`, yang juga dapat digunakan untuk memformat vektor untuk output.

```
>convertm xm(1:10)
```

```
[1,2,3,4,5,6,7,8,9,10]
```

Untuk Latex perintah `tex` dapat digunakan untuk mendapatkan perintah Latex.

```
>tex(&[1,2,3])
```

```
\left[ 1 , 2 , 3 \right]
```

Faktor dan Tabel

Dalam pengantar R ada contoh dengan apa yang disebut faktor.

Berikut ini adalah daftar wilayah dari 30 negara bagian.

```
>austates = ["tas", "sa", "qld", "nsw", "nsw", "nt", "wa", "wa", ...
>"qld", "vic", "nsw", "vic", "qld", "qld", "sa", "tas", ...
>"sa", "nt", "wa", "vic", "qld", "nsw", "nsw", "wa", ...
>"sa", "act", "nsw", "vic", "vic", "act"];
```

Asumsikan, kita memiliki pendapatan yang sesuai di setiap negara bagian.

```
>incomes = [60, 49, 40, 61, 64, 60, 59, 54, 62, 69, 70, 42, 56, ...
>61, 61, 61, 58, 51, 48, 65, 49, 49, 41, 48, 52, 46, ...
>59, 46, 58, 43];
```

Sekarang, kami ingin menghitung rata-rata pendapatan di wilayah tersebut. Menjadi program statistik, R memiliki factor() dan tapply() untuk ini.

EMT dapat melakukannya dengan menemukan indeks wilayah dalam daftar wilayah yang unik.

```
>auterr=sort(unique(austates)); f=indexofsorted(auterr,austates)
```

```
[6, 5, 4, 2, 2, 3, 8, 8, 4, 7, 2, 7, 4, 4, 5, 6, 5, 3,
8, 7, 4, 2, 2, 8, 5, 1, 2, 7, 7, 1]
```

Pada titik itu, kita dapat menulis fungsi loop kita sendiri untuk melakukan sesuatu hanya untuk satu faktor.

Atau kita bisa meniru fungsi tapply() dengan cara berikut.

```
>function map_tappl (i; f$call, cat, x) ...
```

```
u=sort(unique(cat));
f=indexof(u,cat);
return f$(x[nonzeros(f==indexof(u,i))]);
endfunction
```

Ini agak tidak efisien, karena menghitung wilayah unik untuk setiap i , tetapi berhasil.

```
>tapply(auterr,"mean",austates,incomes)
```

```
[44.5, 57.3333333333, 55.5, 53.6, 55, 60.5, 56, 52.25]
```

Perhatikan bahwa ini berfungsi untuk setiap vektor wilayah.

```
>tapply(["act","nsw"],"mean",austates,incomes)
```

```
[44.5, 57.333333333]
```

Sekarang, paket statistik EMT mendefinisikan tabel seperti di R. Fungsi `readtable()` dan `writetable()` dapat digunakan untuk input dan output.

Jadi kita bisa mencetak rata-rata pendapatan negara di wilayah dengan cara yang bersahabat.

```
>writetable(tapply(auterr,"mean",austates,incomes),labc=auterr,wc=7)
```

	act	nsw	nt	qld	sa	tas	vic	wa
	44.5	57.33	55.5	53.6	55	60.5	56	52.25

Kita juga dapat mencoba meniru perilaku R sepenuhnya.

Faktor-faktor tersebut harus dengan jelas disimpan dalam kumpulan dengan jenis dan kategori (negara bagian dan teritori dalam contoh kami). Untuk EMT, kami menambahkan indeks yang telah dihitung sebelumnya.

```
>function makef (t) ...
```

```
## Factor data
## Returns a collection with data t, unique data, indices.
## See: tapply
u=sort(unique(t));
return {{t,u,indexofsorted(u,t)}};
endfunction
```

```
>statef=makef(austates);
```

Sekarang elemen ketiga dari koleksi akan berisi indeks.

```
>statef[3]
```

```
[6, 5, 4, 2, 2, 3, 8, 8, 4, 7, 2, 7, 4, 4, 5, 6, 5, 3,
8, 7, 4, 2, 2, 8, 5, 1, 2, 7, 7, 1]
```

Sekarang kita bisa meniru tapply() dengan cara berikut. Ini akan mengembalikan tabel sebagai kumpulan data tabel dan judul kolom.

```
>function tapply (t:vector, tf, f$:call) ...
```

```
## Makes a table of data and factors
## tf : output of makef()
## See: makef
uf=tf[2]; f=tf[3]; x=zeros(length(uf));
for i=1 to length(uf);
    ind=nonzeros(f==i);
    if length(ind)==0 then x[i]=NAN;
    else x[i]=f$(t[ind]);
    endif;
end;
return {{x,uf}};
endfunction
```

Kami tidak menambahkan banyak jenis pengecekan di sini. Satu-satunya tindakan pencegahan menyangkut kategori (faktor) tanpa data. Tetapi orang harus memeriksa panjang t yang benar dan kebenaran koleksi tf.

Tabel ini dapat dicetak sebagai tabel dengan writetable().

```
>writetable(tapply(incomes,statef,"mean"),wc=7)
```

act	nsw	nt	qld	sa	tas	vic	wa
44.5	57.33	55.5	53.6	55	60.5	56	52.25

Array

EMT hanya memiliki dua dimensi untuk array. Tipe datanya disebut matriks. Akan mudah untuk menulis fungsi untuk dimensi yang lebih tinggi atau pustaka C untuk ini.

R memiliki lebih dari dua dimensi. Dalam R array adalah vektor dengan bidang dimensi. Dalam EMT, vektor adalah matriks dengan satu baris. Itu dapat dibuat menjadi matriks dengan redim().

```
>shortformat; X=redim(1:20,4,5)
```

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20

Ekstraksi baris dan kolom, atau sub-matriks, sangat mirip dengan R.

```
>X[,2:3]
```

2	3
7	8
12	13
17	18

Namun, dalam R dimungkinkan untuk menetapkan daftar indeks spesifik dari vektor ke suatu nilai. Hal yang sama dimungkinkan di EMT hanya dengan loop.

```
>function setmatrixvalue (M, i, j, v) ...
```

```
loop 1 to max(length(i),length(j),length(v))
    M[i#{},j#{}] = v#;
end;
endfunction
```

Kami mendemonstrasikan ini untuk menunjukkan bahwa matriks dilewatkan dengan referensi di EMT. Jika Anda tidak ingin mengubah matriks asli M, Anda perlu menyalinnya ke dalam fungsi.

```
>setmatrixvalue(X,1:3,3:-1:1,0); X,
```

1	2	0	4	5
6	0	8	9	10
0	12	13	14	15
16	17	18	19	20

Perkalian luar dalam EMT hanya dapat dilakukan antar vektor. Ini otomatis karena bahasa matriks. Satu vektor harus menjadi vektor kolom dan yang lainnya vektor baris.

```
> (1:5) * (1:5)'
```

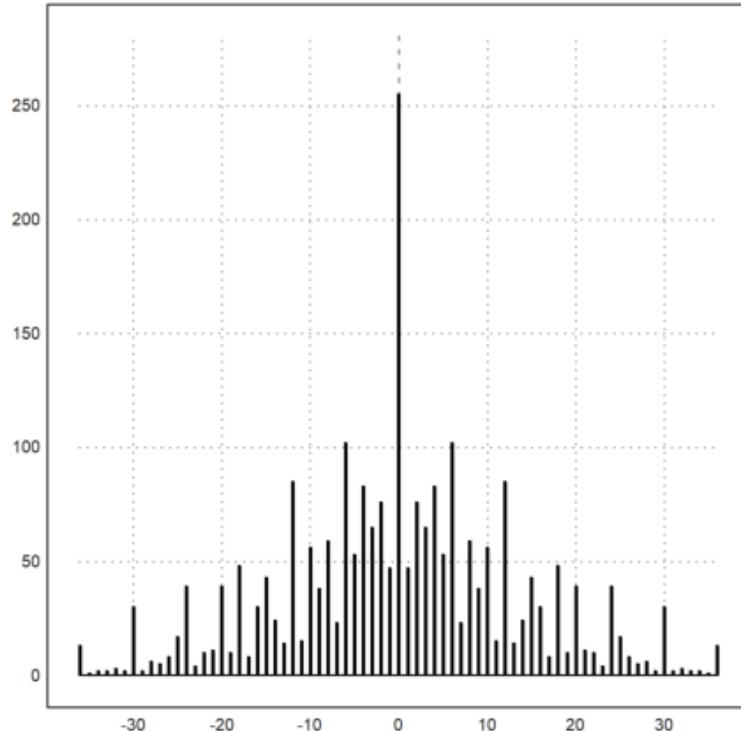
1	2	3	4	5
2	4	6	8	10
3	6	9	12	15
4	8	12	16	20
5	10	15	20	25

Dalam pengantar PDF untuk R ada sebuah contoh, yang menghitung distribusi ab-cd untuk a,b,c,d yang dipilih dari 0 hingga n secara acak. Solusi dalam R adalah membentuk matriks 4 dimensi dan menjalankan table() di atasnya.

Tentu saja, ini dapat dicapai dengan loop. Tapi loop tidak efektif di EMT atau R. Di EMT, kita bisa menulis loop di C dan itu akan menjadi solusi tercepat.

Tapi kita ingin meniru perilaku R. Untuk ini, kita perlu meratakan perkalian ab dan membuat matriks ab-cd.

```
>a=0:6; b=a'; p=flatten(a*b); q=flatten(p-p'); ...
>u=sort(unique(q)); f=getmultiplicities(u,q); ...
>statplot(u,f,"h"):
```



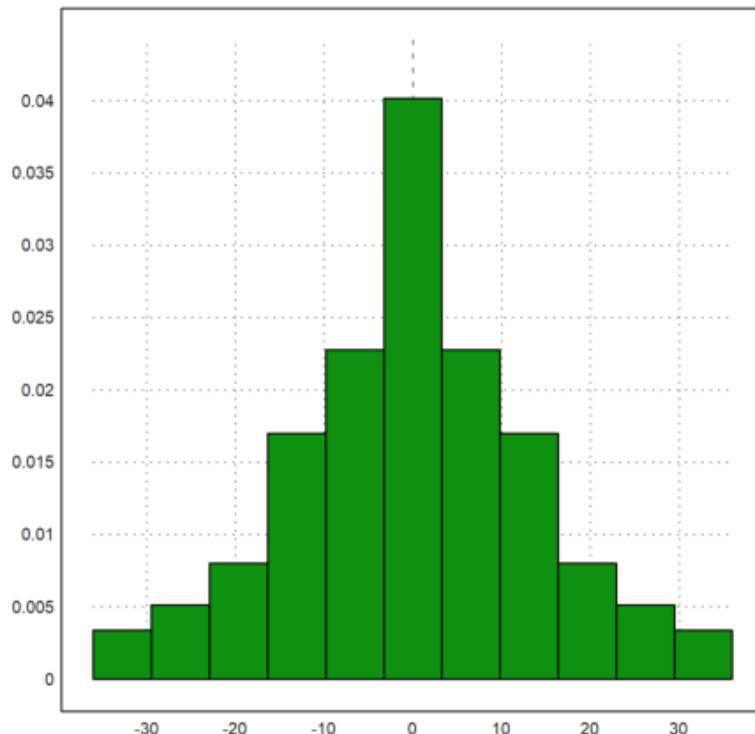
Selain multiplisitas yang tepat, EMT dapat menghitung frekuensi dalam vektor.

```
>getfrequencies(q, -50:10:50)
```

```
[0, 23, 132, 316, 602, 801, 333, 141, 53, 0]
```

Cara paling mudah untuk memplot ini sebagai distribusi adalah sebagai berikut.

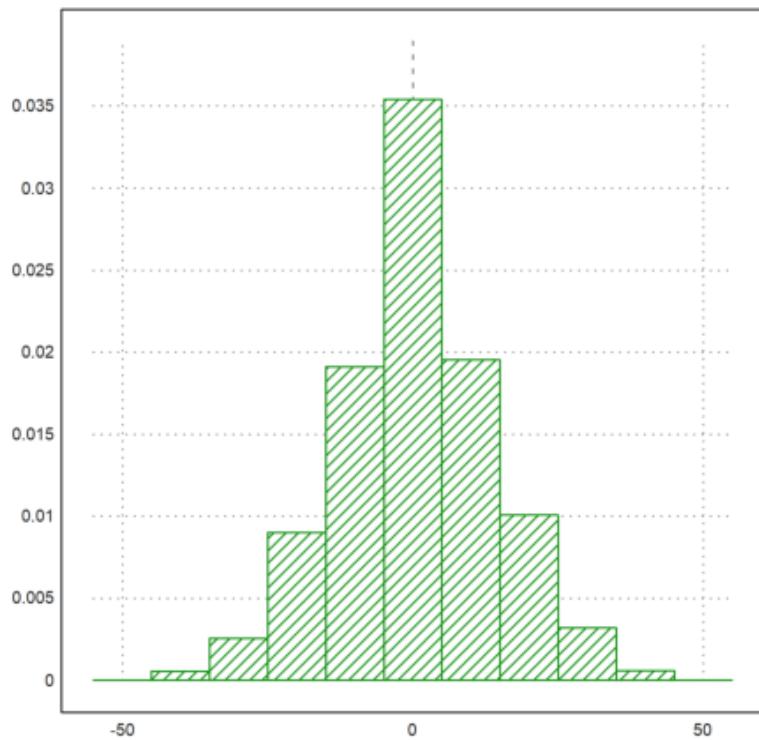
```
>plot2d(q,distribution=11):
```



Tetapi juga dimungkinkan untuk menghitung sebelumnya hitungan dalam interval yang dipilih sebelumnya. Tentu saja, berikut ini menggunakan getfrequencies() secara internal.

Karena fungsi histo() mengembalikan frekuensi, kita perlu menskalakannya sehingga integral di bawah grafik batang adalah 1.

```
>{x,y}=histo(q,v=-55:10:55); y=y/sum(y)/differences(x); ...
>plot2d(x,y,>bar,style="/"):
```



Daftar

EMT memiliki dua macam daftar. Salah satunya adalah daftar global yang dapat diubah, dan yang lainnya adalah jenis daftar yang tidak dapat diubah. Kami tidak peduli dengan daftar global di sini.

Jenis daftar yang tidak dapat diubah disebut koleksi di EMT. Itu berperilaku seperti struktur di C, tetapi elemennya hanya diberi nomor dan tidak diberi nama.

```
>L={ {"Fred", "Flintstone", 40, [1990, 1992] } }
```

```
Fred
Flintstone
40
[1990, 1992]
```

Saat ini elemen tidak memiliki nama, meskipun nama dapat ditetapkan untuk tujuan khusus. Mereka diakses dengan angka.

```
>(L[4])[2]
```

```
1992
```

File Input dan Output (Membaca dan Menulis Data)

Anda akan sering ingin mengimpor matriks data dari sumber lain ke EMT. Tutorial ini memberitahu Anda tentang banyak cara untuk mencapai ini. Fungsi sederhana adalah writematrix() dan readmatrix().

Mari kita tunjukkan cara membaca dan menulis vektor real ke file.

```
>a=random(1,100); mean(a), dev(a),
```

```
0.49815  
0.28037
```

Untuk menulis data ke file, kita menggunakan fungsi writematrix().

Karena pengenalan ini kemungkinan besar berada di direktori, di mana pengguna tidak memiliki akses tulis, kami menulis data ke direktori home pengguna. Untuk notebook sendiri, ini tidak perlu, karena file data akan ditulis ke dalam direktori yang sama.

```
>filename="test.dat";
```

Sekarang kita menulis vektor kolom a' ke file. Ini menghasilkan satu nomor di setiap baris file.

```
>writematrix(a',filename);
```

Untuk membaca data, kita gunakan readmatrix().

```
>a=readmatrix(filename)';
```

dan hapus file ini.

```
>fileremove(filename);  
>mean(a), dev(a),
```

```
0.49815  
0.28037
```

Fungsi writematrix() atau writetable() dapat dikonfigurasi untuk bahasa lain.

Misalnya, jika Anda memiliki sistem Indonesia (titik desimal dengan koma), Excel Anda memerlukan nilai dengan koma desimal yang dipisahkan oleh titik koma dalam file csv (defaultnya adalah nilai yang dipisahkan koma). File "test.csv" berikut akan muncul di folder current Anda.

```
>filename="test.csv"; ...
>writematrix(random(5,3),file=filename,separator=",");
```

Anda sekarang dapat membuka file ini dengan Excel Indonesia secara langsung.

```
>fileremove(filename);
```

Terkadang kita memiliki string dengan token seperti berikut ini.

```
>s1:="f m m f m m m f f f m m f"; ...
>s2:="f f f m m f f";
```

Untuk tokenize ini, kita mendefinisikan vektor token.

```
>tok:=[ "f", "m" ]
```

f

m

Kemudian kita dapat menghitung berapa kali setiap token muncul dalam string, dan memasukkan hasilnya ke dalam tabel.

```
>M:=getmultiplicities(tok,strtokens(s1))_ ...
>  getmultiplicities(tok,strtokens(s2));
```

Tulis tabel dengan header token.

```
>writetable(M,labc=tok,labr=1:2,wc=8)
```

	f	m
1	6	7
2	5	2

Untuk statika, EMT dapat membaca dan menulis tabel.

```
>file="test.dat"; open(file,"w"); ...
>writeln("A,B,C"); writematrix(random(3,3)); ...
>close();
```

Filenya terlihat seperti ini.

```
>printfile(file)
```

```
A, B, C
0.7003664386138074, 0.1875530821001213, 0.3262339279660414
0.5926249243193858, 0.1522927283984059, 0.368140583062521
0.8065535209872989, 0.7265910840408142, 0.7332619844597152
```

Fungsi `readtable()` dalam bentuknya yang paling sederhana dapat membaca ini dan mengembalikan kumpulan nilai dan baris judul.

```
>L=readtable(file,>list);
```

Koleksi ini dapat dicetak dengan `writetable()` ke notebook, atau ke file.

```
>writetable(L,wc=10,dc=5)
```

A	B	C
0.70037	0.18755	0.32623
0.59262	0.15229	0.36814
0.80655	0.72659	0.73326

Matriks nilai adalah elemen pertama dari L. Perhatikan bahwa `mean()` dalam EMT menghitung nilai rata-rata dari baris matriks.

```
>mean(L[1])
```

```
0.40472
0.37102
0.75547
```

File CSV

Pertama, mari kita menulis matriks ke dalam file. Untuk output, kami membuat file di direktori kerja saat ini.

```
>file="test.csv"; ...
>M=random(3,3); writematrix(M,file);
```

Berikut adalah isi dari file ini.

```
>printfile(file)
```

```
0.8221197733097619, 0.821531098722547, 0.7771240608094004
0.8482947121863489, 0.3237767724883862, 0.6501422353377985
0.1482301827518109, 0.3297459716109594, 0.6261901074210923
```

CSV ini dapat dibuka pada sistem bahasa Inggris ke Excel dengan klik dua kali. Jika Anda mendapatkan file seperti itu di sistem Jerman, Anda perlu mengimpor data ke Excel dengan memperhatikan titik desimal.

Tetapi titik desimal juga merupakan format default untuk EMT. Anda dapat membaca matriks dari file dengan readmatrix().

```
>readmatrix(file)
```

```
0.82212 0.82153 0.77712
0.84829 0.32378 0.65014
0.14823 0.32975 0.62619
```

Dimungkinkan untuk menulis beberapa matriks ke satu file. Perintah open() dapat membuka file untuk ditulis dengan parameter "w". Standarnya adalah "r" untuk membaca.

```
>open(file, "w"); writematrix(M); writematrix(M'); close();
```

Matriks dipisahkan oleh garis kosong. Untuk membaca matriks, buka file dan panggil readmatrix() beberapa kali.

```
>open(file); A=readmatrix(); B=readmatrix(); A==B, close();
```

```
1 0 0
0 1 0
0 0 1
```

Di Excel atau spreadsheet serupa, Anda dapat mengekspor matriks sebagai CSV (nilai yang dipisahkan koma). Di Excel 2007, gunakan "simpan sebagai" dan "format lain", lalu pilih "CSV". Pastikan, tabel saat ini hanya berisi data yang ingin Anda ekspor.

Berikut adalah contoh.

```
>printfile("excel-data.csv")
```

```
Could not open the file
excel-data.csv
for reading!
Try "trace errors" to inspect local variables after errors.
printfile:
  open(filename, "r");
```

Seperti yang Anda lihat, sistem Jerman saya menggunakan titik koma sebagai pemisah dan koma desimal. Anda dapat mengubah ini di pengaturan sistem atau di Excel, tetapi tidak perlu membaca matriks ke dalam EMT.

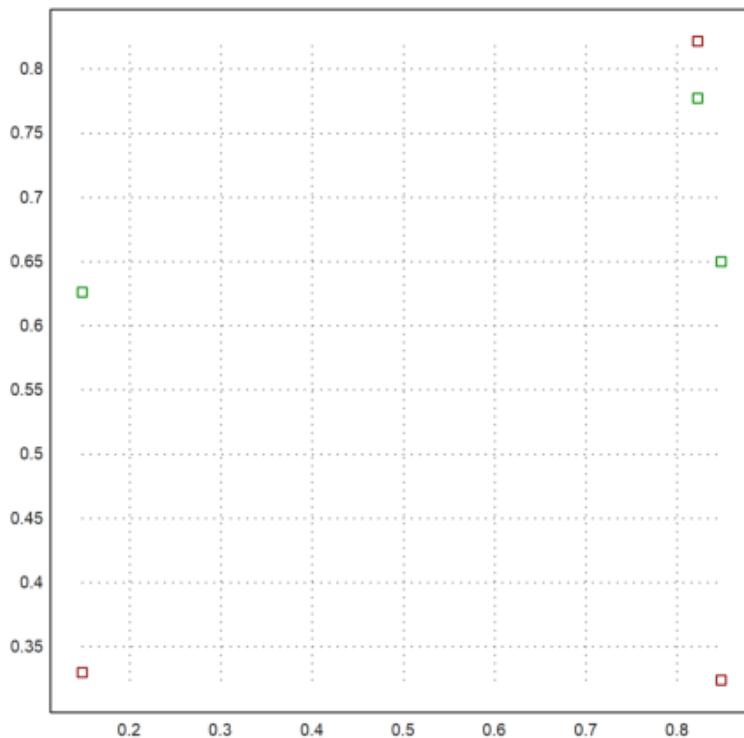
Cara termudah untuk membaca ini ke dalam Euler adalah readmatrix(). Semua koma diganti dengan titik dengan parameter >comma. Untuk CSV bahasa Inggris, cukup abaikan parameter ini.

```
>M=readmatrix("excel-data.csv",>comma)
```

```
Could not open the file
excel-data.csv
for reading!
Try "trace errors" to inspect local variables after errors.
readmatrix:
  if filename<>"" then open(filename, "r"); endif;
```

Mari kita plot ini.

```
>plot2d(M' [1],M' [2:3],>points,color=[red,green]'):
```



Ada cara yang lebih mendasar untuk membaca data dari file. Anda dapat membuka file dan membaca angka baris demi baris. Fungsi getvectorline() akan membaca angka dari baris data. Secara default, ia mengharapkan titik desimal. Tapi itu juga bisa menggunakan koma desimal, jika Anda memanggil setdecimaldot(",") sebelum Anda menggunakan fungsi ini.

Fungsi berikut adalah contoh untuk ini. Ini akan berhenti di akhir file atau baris kosong.

```
>function myload (file) ...
```

```
open(file);
M=[];
repeat
    until eof();
    v=getvectorline(3);
    if length(v)>0 then M=M_v; else break; endif;
end;
return M;
close(file);
endfunction
```

```
>myload(file)
```

0.82212	0	0.82153	0	0.77712
0.84829	0	0.32378	0	0.65014
0.14823	0	0.32975	0	0.62619

Dimungkinkan juga untuk membaca semua angka dalam file itu dengan getvector().

```
>open(file); v=getvector(10000); close(); redim(v[1:9],3,3)
```

0.82212	0	0.82153
0	0.77712	0.84829
0	0.32378	0

Jadi sangat mudah untuk menyimpan vektor nilai, satu nilai di setiap baris dan membaca kembali vektor ini.

```
>v=random(1000); mean(v)
```

0.50303

```
>writematrix(v',file); mean(readmatrix(file)')
```

0.50303

Menggunakan Tabel

Tabel dapat digunakan untuk membaca atau menulis data numerik. Sebagai contoh, kami menulis tabel dengan header baris dan kolom ke file.

```
>file="test.tab"; M=random(3,3); ...
>open(file,"w"); ...
>writetable(M,separator=",",labc=[ "one", "two", "three"]); ...
>close(); ...
>printfile(file)
```

one,two,three		
0.09,	0.39,	0.86
0.39,	0.86,	0.71
0.2,	0.02,	0.83

Ini dapat diimpor ke Excel.

Untuk membaca file dalam EMT, kami menggunakan readtable().

```
>{M,headings}=readtable(file,>clabs); ...
>writetable(M,labc=headings)
```

one	two	three
0.09	0.39	0.86
0.39	0.86	0.71
0.2	0.02	0.83

Menganalisis Garis

Anda bahkan dapat mengevaluasi setiap baris dengan tangan. Misalkan, kita memiliki garis dengan format berikut.

```
>line="2020-11-03,Tue,1'114.05"
```

2020-11-03, Tue, 1'114.05

Pertama kita dapat menandai garis.

```
>vt=strtoks(line)
```

2020-11-03
Tue
1'114.05

Kemudian kita dapat mengevaluasi setiap elemen garis menggunakan evaluasi yang sesuai.

```
>day(vt[1]), ...
>indexof(["mon","tue","wed","thu","fri","sat","sun"],tolower(vt[2])), ...
>strrepl(vt[3], "'", "")()
```

7.3816e+05
2
1114

Menggunakan ekspresi reguler, dimungkinkan untuk mengekstrak hampir semua informasi dari baris data.

Asumsikan kita memiliki baris berikut dokumen HTML.

```
>line="<tr><td>1145.45</td><td>5.6</td><td>-4.5</td><tr>"
```

```
<tr><td>1145.45</td><td>5.6</td><td>-4.5</td><tr>
```

Untuk mengekstrak ini, kami menggunakan ekspresi reguler, yang mencari

- kurung tutup >,
- string apa pun yang tidak mengandung tanda kurung dengan

sub-pertandingan "(...)".

- braket pembuka dan penutup menggunakan solusi terpendek,
- lagi string apa pun yang tidak mengandung tanda kurung,
- dan kurung buka <.

Ekspresi reguler agak sulit dipelajari tetapi sangat kuat.

```
>{pos,s,vt}=strxfind(line,>([<>]+)<.+?>([<>]+)<" );
```

Hasilnya adalah posisi kecocokan, string yang cocok, dan vektor string untuk sub-pertandingan.

```
>for k=1:length(vt); vt[k](); end;
```

```
1145.5
5.6
```

Berikut adalah fungsi, yang membaca semua item numerik antara <td> dan </td>.

```
>function readtd (line) ...
```

```
v=[]; cp=0;
repeat
    {pos,s,vt}=strxfind(line,<td.*?>(.+?)</td>,cp);
    until pos==0;
    if length(vt)>0 then v=v|vt[1]; endif;
    cp=cp+strlen(s);
end;
return v;
endfunction
```

```
>readtd(line+"<td>non-numerical</td>")
```

```
1145.45
5.6
-4.5
non-numerical
```

Membaca dari Web

Situs web atau file dengan URL dapat dibuka di EMT dan dapat dibaca baris demi baris. Dalam contoh, kami membaca versi saat ini dari situs EMT. Kami menggunakan ekspresi reguler untuk memindai "Versi ..." dalam sebuah judul.

```
>function readversion () ...
```

```
urlopen("http://www.euler-math-toolbox.de/Programs/Changes.html");
repeat
  until urleof();
  s=urlgetline();
  k=strfind(s,"Version ",1);
  if k>0 then substring(s,k,strfind(s,<,k)-1), break; endif;
end;
urlclose();
endfunction
```

```
>readversion
```

Version 2022-05-18

Input dan Output Variabel

Anda dapat menulis variabel dalam bentuk definisi Euler ke file atau ke baris perintah.

```
>writevar(pi,"mypi");
```

```
mypi = 3.141592653589793;
```

Untuk pengujian, kami membuat file Euler di direktori kerja EMT.

```
>file="test.e"; ...
>writevar(random(2,2), "M", file); ...
>printfile(file, 3)
```

```
M = [ ..
0.5991820585590205, 0.7960280262224293;
0.5167243983231363, 0.2996684599070898];
```

Kita sekarang dapat memuat file. Ini akan mendefinisikan matriks M.

```
>load(file); show M,
```

```
M =
0.59918   0.79603
0.51672   0.29967
```

Omong-omong, jika writevar() digunakan pada variabel, itu akan mencetak definisi variabel dengan nama variabel ini.

```
>writevar(M); writevar(inch$)
```

```
M = [ ..
0.5991820585590205, 0.7960280262224293;
0.5167243983231363, 0.2996684599070898];
inch$ = 0.0254;
```

Kita juga bisa membuka file baru atau menambahkan file yang sudah ada. Dalam contoh kami menambahkan ke file yang dihasilkan sebelumnya.

```
>open(file, "a"); ...
>writevar(random(2,2), "M1"); ...
>writevar(random(3,1), "M2"); ...
>close();
>load(file); show M1; show M2;
```

```
M1 =
0.30287   0.15372
0.7504    0.75401
M2 =
0.27213
0.053211
0.70249
```

Untuk menghapus file apa pun, gunakan fileremove().

```
>fileremove(file);
```

Vektor baris dalam file tidak memerlukan koma, jika setiap angka berada di baris baru. Mari kita buat file seperti itu, menulis setiap baris satu per satu dengan writeln().

```
>open(file, "w"); writeln("M = [ "); ...
>for i=1 to 5; writeln(""+random()); end; ...
>writeln("]"); close(); ...
>printfile(file)
```

```
M = [
0.344851384551
0.0807510017715
0.876519562911
0.754157709472
0.688392638934
];
```

```
>load(file); M
```

```
[0.34485, 0.080751, 0.87652, 0.75416, 0.68839]
```

catatan : ketika mengenter perintah-perintah diatas ternyata hasil yang didapatkan berbeda-beda

Latihan soal

Nomor 1

Carilah rata-rata dan standar deviasi beserta plot dari data berikut

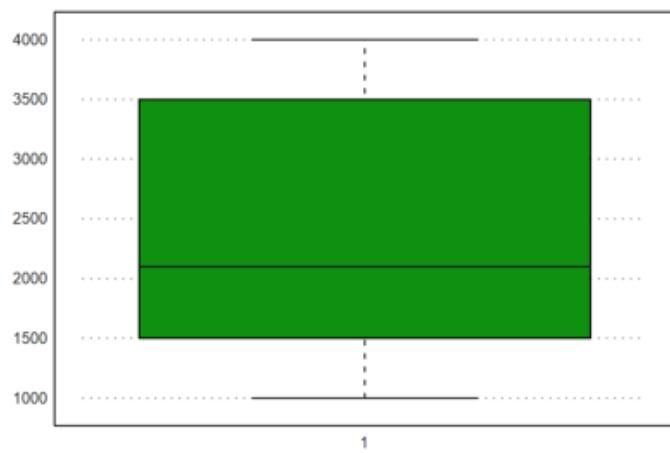
X = 1000,1500,1700,2500,3500,4000

```
>X=[1000,1500,1700,2500,3500,4000]; ...
>mean(X), dev(X),
```

2366.7

1186

```
>aspect(1.5); boxplot(X):
```



Nomor 2

Misalkan diberikan data skor hasil statistika dari 14 orang mahasiswa sebagai berikut:
50,92,68,72,84,80,96,64,70,48,88,66,56,84

Tentukan rata-rata dari data tersebut!

```
>X=[ 62, 65, 58, 90, 75, 79, 82, 91, 75, 75, 75, 75, 95]
```

```
[ 62, 65, 58, 90, 75, 79, 82, 91, 75, 75, 75, 75, 95]
```

```
>mean(X)
```

76.833